EUROLAN 2007 Summer School

Alexandru Ioan Cuza University of Iaşi

Proceedings of the Doctoral Consortium

Iaşi, July 30 – August 2, 2007

Publishing House of the Alexandru Ioan Cuza University of Iaşi
EUROLAN 2007 Summer School

Alexandru Ioan Cuza University of Iași

Proceedings of the *Doctoral Consortium*

Iași, July 30 – August 2, 2007

**Editors**

Corina Forăscu

Oana Postolache

Georgiana Pușcașu

Cristina Vertan

Iași, 2007
Program Committee

Galia Angelova, IPP-BAS, Bulgarian Academy of Sciences, Bulgaria
Carlos Areces, LORIA, INRIA, Nancy, France
Patrick Blackburn, LORIA, INRIA, Nancy, France
Antonio Branco, University of Lisbon, Portugal
Dan Cristea, University Al.I. Cuza of Iaşi, Romania
Corina Forăscu, University Al.I. Cuza of Iaşi, Romania
Claire Gardent, LORIA, CNRS, Nancy, France
Maria Georgescul, ISSCO/TIM/ETI, University of Geneva, Switzerland
Jerry Hobbs, ISI – University of Southern California, US
Nancy Ide, Vassar College, USA
Diana Inkpen, University of Ottawa, Canada
Alex Kelling, Swiss Federal Institute of Technology, Zürich, Switzerland
Zornitsa Kozareva, University of Alicante, Spain
Lothar Lemnitzer, Tübingen University, Germany
Rada Mihalcea, University of North Texas, US
Paola Monachesi, Uil-OTS, Utrecht Univ., The Netherlands
Enrico Motta, Open University, UK
Vivi Năstase, EML Research gGmbH, Germany
Nicolas Nicolov, Umbria Inc., Boulder, US
Constantin Orăsan, University of Wolverhampton, UK
Petya Osenova, IPP-BAS, Bulgarian Academy of Sciences, Bulgaria
Ted Pedersen, University of Minnesota in Duluth, US
Livia Polanyi, Powerset Inc., US
Oana Postolache, ISI – University of Southern California, US
Adam Przepiórkowski, Polish Academy of Sciences, Poland
Georgiana Puşcaşu, University of Wolverhampton, UK
Mike Rosner, University of Malta
Violeta Seretan, University of Geneva, Switzerland
Kiril Simov, IPP-BAS, Bulgarian Academy of Sciences, Bulgaria
Oliviero Stock, Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy
Carlo Strapparava, Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy
Amalia Todiraşcu, Marc Bloch University, Strasbourg, France
Dan Tufiş, Romanian Academy, Bucharest, Romania
Dusko Vitas, University of Belgrade, Serbia
Cristina Vertan, University of Hamburg, Germany
Walter von Hahn, University of Hamburg, Germany
Jan Wiebe, Dept. of Computer Science, Univ. of Pittsburgh, US
Michael Zock, LIF-CNRS, Marseille, France
**Foreword**

The Doctoral Consortium at EUROLAN-2007 provides an opportunity for graduate students (PhD and MSc heading towards PhD studies) investigating topics in Computational Linguistics and Natural Language Processing to present their current work and receive constructive feedback and guidance on future research, both from the general audience and the invited lecturers at the Summer School.

We received 22 submissions and after the review process we accepted 12 full-papers (to be orally presented) and 5 five poster-papers (included in these Proceedings and presented as posters during the event). The papers are grouped under 4 main, general chapters, according to their topic.

It is the first time in the history of the biennial EUROLAN Summer Schools – already at its 8th edition – to have a satellite event dedicated especially to young PhD researchers. Judging after the echo we received, the research community is well-open to such events. Next to the 19 authors (out of 21) that will present their papers, about 40 other participants to the Summer School will take part to the Doctoral Consortium.

We would like to thank all the authors who submitted papers for their participation and for their prompt modification of their submissions according to the reviewers comments. We express our deepest gratitude to all Program Committee members for their thorough reviews. Last but not least, this volume wouldn’t have been possible without the support obtained from the Romanian Ministry of Research and the cooperation of its representatives. We thank all the participants for their involvement, feedback and contributions to the Doctoral Consortium.

The editors
# Table of Contents

Foreword ................................................................................................................................. 1

1 Sentiments and Affects ......................................................................................................... 3
   EmoTag: Automated Mark Up of Affective Information in Texts ....................................... 5
   Virginia Francisco, Raquel Hervás
   Sentiment and near-synonymy: do they go together? ......................................................... 13
   Mary Gardiner
   Sentiment zones approach to extracting a training corpus in unsupervised sentiment classification .......... 21
   Taras Zagibalov

2 Models of Language .......................................................................................................... 29
   A mereology-based general linearization model for surface realization
   Ciprian-Virgil Gerstenberger
   Improving target language modeling techniques for statistical machine translation ................. 39
   Maxim Khalilov
   Using a Single Framework for Computational Modeling of Linguistic Similarity for Solving Many NLP Problems .......................................................... 46
   Anil Kumar Singh, Harshit Surana
   RomGen - A Romanian morphological generator, essential component of an English-to-Romanian MT system . 54
   Irimia Elena

3 Applications in Natural Language Processing .................................................................. 59
   Discovering grammar rules for Automatic Extraction of Definitions ................................. 61
   Claudia Borg
   SketchNet: Knowledge Based Word Sense Disambiguation .............................................. 69
   Alan Duca
   F-structure: Derivation-based construction and constraints ................................................ 77
   Miltiadis Kokkonidis
   Document Editing through Speech .................................................................................... 85
   Dawn Borg Costanzi
   Statistical Machine Translation for Mobile Devices .......................................................... 89
   Alexandru Ceausu
   Unsupervised Cohesion Based Text Segmentation ............................................................ 93
   Costin-Gabriel Chiru

4 Discourse Processing .......................................................................................................... 97
   Textual Entailment .............................................................................................................. 99
   Adrian Iftene
   Discourse parsing using NLP architectures ...................................................................... 107
   Ionuţ Cristian Pistol
   Pronominal Anaphora Generation in Human-Computer Dialogue, Using a Semantic-Pragmatic Approach ...... 114
   Vladimir Popescu\textsuperscript{1, 2}
   Anaphora Resolution for the ARE Competition .................................................................. 122
   Gabriela Pavel, Iustin Dornescu, Alex Moruz
1 Sentiments and Affects
EmoTag: Automated Mark Up of Affective Information in Texts.

Virginia Francisco
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
virginia@fdi.ucm.es

Raquel Hervás
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid, Spain
raquelhb@fdi.ucm.es

Abstract
This paper presents an approach to automated mark up of affective information in texts. The approach considers in parallel two possible representations of emotions: as emotional categories and emotional dimensions. For each representation, a corpus of example texts previously annotated by human evaluators is mined for an initial assignment of emotional features to words. This results in a List of Emotional Words (LEW) which becomes a useful resource for later automated mark up. EmoTag employs for the actual assignment of emotional features a combination of the LEW resource, the ANEW word list, WordNet for knowledge-based expansion of words not occurring in either and an ontology of emotional categories.

1 Introduction
The task of annotating text with specific labels indicating its emotional content or inclination is fundamental for any attempt to make computer interfaces respond in some way to the affective nature of the content they are handling. This is particularly true for research which attempts to produce synthesized voice with different emotional states, but it may also be applicable in other contexts, such as multimodal presentation, where colors, typography or similar means can be used to convey emotion.

In NLP is important to know the connection between emotions and words and linguistic structures in order to drive plot generation in automatic narrative generation. There are programs as MEXICA (Pérez y Pérez and Sharples, 2001) which employ the emotional links between characters to retrieve possible logical actions to continue the story while developing a narration. In this type of programs it will be interesting retrieve the words and the linguistic structures which adapt to the emotion that the story is trying to express with the actions retrieved to continue the story in a emotional way.

Emotions are not an easy phenomenon; there are a lot of factors that contribute to the generation of emotions. For Izard (Izard, 1971) a good definition of the word emotion must take into account: the conscious feeling of the emotion, the processes that appear in the nervous system and in the brain and the expressive models of the emotion.

There are a lot of theories about how many emotions there actually are. In these theories the number of emotions varies from two up to infinity (Ortony and Turner, 1990). There are two main approaches related to the number of emotions: emotional categories and emotional dimensions. The first approach is based on the use of emotion-denoting words (Cowie and Cornelius, 2003). The second approach is that there are two or three dimensions (Dietz and Lang, 1999) which represent the essential aspects of emotional concepts. Common labels for these dimensions include: evaluation (positive / negative) and activation (active / passive) which are the main dimensions; sometimes they are completed with the power dimension (dominant / submissive).

Many psychologists have claimed that certain emotions are more basic than others. Plutchik's (1980) postulates that there is a small number of basic, primary, or prototype emotions (anger, anticipation, disgust, joy, fear, sadness and surprise). All other emotions are mixed or derivative states.
Plutchik states that all emotions vary in their degree of similarity to one another and that each emotion can exist in varying degrees of intensity or levels of arousal. Ekman (1992) has focused on a set of six basic emotions that have associated facial expressions: anger, disgust, fear, joy, sadness and surprise. Those emotions are distinctive, among other properties, by the facial expression characteristic to each one. Izard (1977) determines that the basic emotions are anger, contempt, disgust, distress, guilt, interest, joy, shame and surprise. The OCC Model (Ortony et al, 1988) has established itself as the standard model for emotional synthesis. It presents 22 emotional categories: pride - shame, admiration - reproach, happy - resentment, gloating - pity, hope - fear, joy - distress, satisfaction - fear-confirmed, relief - disappointment, gratification - remorse, gratitude - anger and love - hate. OCC Model considers that categories are based on valence reactions to situations constructed as: goal relevant actions and attractive or unattractive objects. Parrot (2001) presents a deeper list of emotions, where emotions were categorized into a short tree structure, this structure has three levels: primary emotions, secondary emotions and tertiary emotions. As primary emotions Parrot presents: love, joy, surprise, anger, sadness and fear.

The aim of this work is to present a system for the automated mark up of affective information in texts, EmoTag (Francisco and Gervás, 2006b). The last section discusses some ideas for future work which will improve the results obtained with the first approach.

2 Resources employed by EmoTag

This section presents a brief review of the existing resources used by EmoTag.

The Affective Norms for English Words (ANEW) (Bradley and Lang, 1999) is a set of normative emotional ratings for a large number of words in the English language. The goal is to have a set of verbal materials rated in terms of pleasure, arousal, and dominance. This data base of emotional words is content independent, a set of words have been showed to subjects and each word is rated giving a value for each emotional dimension. We use ANEW for marking up texts with emotional dimensions in order to look for the words which does not appear in our list of emotional words (LEW list) which is content dependent.

WordNet is a semantic lexicon for the English language. It groups English words into sets of synonyms called synsets. We used WordNet for knowledge-based expansion of words not occurring in LEW list (or ANEW list in the case of emotional dimensions). By means of WordNet we obtained synonyms, antonyms and hypernyms (words related with the original word).

Shallow Parsing Techniques are used for syntax analysis. This type of syntax analysis employs the binary relations between lexical units. There are a lot of automatic dependency analyzers for different languages: English, French, Swedish... The most successful is MINIPAR (Lin, 1998) which is used by EmoTag (Francisco and Gervás, 2006a) to determine the scope of negations appearing in the sentences, in order to take their effect into account. Nowadays there are no dependency analyzers in Spanish so we are studying the use of tools as MaltParser 1 in order to get a Spanish dependency analyzer automatically.

A POS Tagger marks up the words in a text with its corresponding part of speech, based on both its definition as well as its context. We used qtag 2 for English and Tree Tagger 3 for Spanish.

A Stemmer reduces inflected (or sometimes derived) words to their stem, base or root form. The stem does not need to be identical to the morphological root of the word, it is sufficient that related words map to the same stem. We need a stem in order to group the related words in our LEW list, that have the same emotional content. In the case of English we have used the stem given by MINIPAR and in the Spanish case we have used Snowball Spanish Stemmer 4.

3 Implemented Resources

This section presents a brief review of the resources that we have implemented in order to mark up texts with emotions.

---

1 http://w3.msi.vxu.se/~nivre/research/MaltParser.html
2 http://www.english.bham.ac.uk/staff/omason/software/qtag.html
3 http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html
4 www.snowball.tartarus.org
3.1 Ontology of Emotions (OntoEmotions)

We have developed an ontology of emotional categories (Francisco et al, 2007). They are structured in a taxonomy that covers from basic emotions to the most specific emotional categories. This ontology is based on the emotional structures mentioned in Section 1, as basic emotions we have: sadness, happiness, surprise, fear and anger. We have adapted the Parrot model to these basic emotions, and we have integrated in this model all the emotions which appear in other models. Finally we have added all the emotion-denoting words of the English and Spanish language.

OntoEmotions is an ontology of emotional categories, that is emotion-denoting words such as happy, sad, fear… not an ontology of words and their relation with emotions as WordNet Affect.

Our ontology has two root concepts:

- Emotion: The root for all the emotional concepts which are used to refer to emotions. Each of the emotional concepts are subclasses of the root concept Emotion. Some examples of these subclasses are: Happiness, Sadness, Fear, Envy…

- Word: The root for the emotion-denoting words, the specific words which each language provides for denoting emotions. Our ontology is currently available for two different languages: English and Spanish. In order to classify the words into their corresponding language the root concept Word has two subclasses: EnglishWord and SpanishWord.

As instances of the EnglishWord and SpanishWord subclasses there are emotion-denoting words, which are all the words used for denoting Surprise, Happiness, Indignation, Horror… Each of these instances has two parents: a concept from the Emotion hierarchy (which indicates the type of abstract emotion denoted by the word) and a concept from the Word hierarchy (which indicates the language of the word).

Figure 1 shows a fragment of the ontology. In this fragment it can be seen how the words are related both to one emotional concept and to one word concept, for example the word cheerfulness is an instance of the emotional concept Happiness at the same time it is an instance of the word concept EnglishWord, which means that cheerfulness is an English word for denoting the emotion Happiness.

![Figure 1. Fragment of the emotional ontology.](image)

Given the semantics we have chosen for our ontology, two instances of the Word-concept can be considered to be synonyms if they are also instances of the same single Emotion-concept from the parallel Emotion subhierarchy. For example, in the figure above, we can find that the words amazement and astonishment are synonyms because they are both instances of the Emotion-concept Amazement.

From a given emotion-denoting word by means of our ontology we obtain the direct emotional concept associated to it as well as the more general emotional concept related to the direct emotional concept. We can get, too, the synonyms for an emotional word and the corresponding word in other language. For example, given the emotional word grief, we have Grief as direct emotional concept, Distress, Sadness and Emotion as general emotional concepts, Agonía as Spanish translation and agony, anguish and sorrow as antonyms.

3.2 List of Emotional Words (LEW List)

Our method for annotating text with emotions relies on a dictionary of word to emotion assignments (Francisco and Gervás, 2006b). This is obtained from a corpus of human evaluated texts by applying language analysis techniques. Similar techniques are later applied to assign emotions to sentences from the assignments for the words that compose them.

If we want to obtain a program that marks up texts with emotions, as a human would, we first need a corpus of marked-up texts in order to analyze and obtain a set of key words which we will
use in the mark up process. Each of the texts which forms part of the corpus may be marked by more than one person because assignment of emotions is a subjective task so we have to avoid “subjective extremes”. In order to do that we obtain the emotion assigned to a phrase as the average of the mark-up provided by fifteen evaluators. Therefore the process of obtaining the list of emotional words involves two different phases: evaluation method, several people mark up some texts from our corpus; extraction method, from the mark-up texts we obtain the list of emotional words.

First we have to decide which texts are going to be part of our corpus. We decide to focus the effort on a very specific domain: fairy tales. This decision was taken mainly because generally fairy tales are intended to help children understand better their feelings, and they usually involve instances of the emotions that most children experiment on their way to maturity: happiness, sadness, anger…

Once the domain of the corpus’ texts is established, the set of specific tales that we are going to work with must be selected. We have selected eight tales, every one of them popular tales with different lengths (altogether they result in 10,331 words and 1,084 sentences), in English and Spanish. The eight tales are marked up with emotional categories and emotional dimensions. We provide a list of different emotions in order to help the evaluators in the assignment of emotional categories and the SAM standard which can be seen in the Figure 2 in order to help them in the assignment of values for each dimension.

Based on the tales marked up by the evaluators we obtain two data base of words and their relation to emotional categories and emotional dimensions, one in English and other in Spanish.

Firstly we have to obtain the emotion mostly assigned to each sentence by the evaluators, in order to do that we distinguishes between emotional dimensions and emotional categories:

- Emotional dimensions: In order to get the reference value for each sentence in the text we get the average value for each of the emotional dimensions (evaluation, activation and power).

- Emotional categories: In order to get the reference value we carry out the following process: if at least half of the evaluators agree on the assignment of one emotion, this emotion is taken as the reference value for the sentence. In other case, we group the emotions in levels (according to the level of the emotional concept they refer to) then we obtain the related concepts for the emotion with the lower level. Once these new concepts are added to their corresponding levels, if we have any emotion supported by at least half of the evaluators we take it as the reference value. If two emotions supported by most of the evaluators we get the emotion with a lower level. In other case, we obtain the related concepts for the emotion with the next lower level and repeat this step in ascending order until we have an emotion supported by at least half of the evaluators.

Once we have the reference value for each sentence we carry out the following process in order to get the LEW list. Firstly we split the text into sentences which are processed with MINIPAR in order to obtain the words affected by a negation and with a tagger which assigns a part-of-speech tag to each word in a text. Every sentence is divided into words and with every word and its label we carry out the following process:

- Discard the words with a label which belong to our list of stop POS tags (conjunctions, numbers, determiners, existential there, prepositions…).

- Obtain the stem of the words.
• Insert the pair stem - label into the LEW list with its corresponding emotional value in the case of words not affected by negation and the opposite emotional value in the case of words affected by negation.

Once all the tales have been processed we extend our list with synonyms and antonyms which are look up in WordNet. For inserting related words into the database, the same emotional values of the original word are used in the case of synonyms and the opposite values are used in the case of antonyms.

For emotional dimensions LEW list stores for each pair (word's stem, label) the average value of activation, evaluation and power of that pair in all the analyzed texts. In the Table 1 it can be seen a fragment of the English LEW list for emotional dimensions:

<table>
<thead>
<tr>
<th>Word’s Stem</th>
<th>Label</th>
<th>Act.</th>
<th>Eval.</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>Adj.</td>
<td>6</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Taste</td>
<td>Adj.</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Sword</td>
<td>Noun</td>
<td>7,3</td>
<td>5</td>
<td>5,5</td>
</tr>
</tbody>
</table>

Table 1. Fragment of the LEW list for emotional dimensions.

For emotional categories LEW list stores for each pair (word's stem, label) the probability of this pair of been indicating of one of the categories. In the Table 2 it can be seen a fragment of the English LEW list for emotional categories:

<table>
<thead>
<tr>
<th>Stem</th>
<th>Label</th>
<th>Grief</th>
<th>Sad</th>
<th>Happy</th>
<th>Neutral</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misery</td>
<td>Noun</td>
<td>20%</td>
<td>20%</td>
<td>0%</td>
<td>0%</td>
<td>...</td>
</tr>
<tr>
<td>Death</td>
<td>Noun</td>
<td>43%</td>
<td>7%</td>
<td>0%</td>
<td>29%</td>
<td>...</td>
</tr>
<tr>
<td>Dark</td>
<td>Noun</td>
<td>20%</td>
<td>40%</td>
<td>0%</td>
<td>0%</td>
<td>...</td>
</tr>
<tr>
<td>Marry</td>
<td>Adj.</td>
<td>0%</td>
<td>0%</td>
<td>75%</td>
<td>25%</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2. Fragment of the LEW list for emotional categories.

4 EmoTag

Our process classifies English and Spanish sentences into emotions (emotional categories and emotional dimensions). The first step is to perform sentence detection and tokenization in order to carry out our process based in the relation between words and different emotions. We carry out the process in the following way: we obtain by means of MINIPAR the words affected by negations and the POS tag for every word in the sentence, based on these tags and the stem of the words we decide the emotion of the sentence in the following way:

• If the tag associated with the word is in our label stop list we leave it out in other case we obtain the stem of the word.

• We look up the pair stem - label in the LEW list, if the word is present we assign to it the probability of carrying the emotions we are studying in the case of emotional categories or the value for the emotional dimensions in other case. If the word is affected by a negation we reverse the probability in the case of emotional dimensions and the value of each dimension in the case of emotional dimensions. Based on these emotional values of the words we obtain the final emotion of the sentence.

• If the word is not in the LEW list, in the case of emotional dimensions we look up it in the ANEW list.

• If the word is not in the LEW list in the case of emotional categories, and is not in the LEW list or the ANEW list in the case of emotional dimensions we look up the hypernyms of the word in WordNet, and look them up in the LEW list (and in the ANEW list in the case of emotional dimensions). The first appearance of a hypernym is taken and the emotional content associated to our original word and the new word is inserted in the LEW list for subsequent occurrences of these words in our tales.

• If none of the hypernyms appear in the LEW list we leave out the word and it does not take part in the mark up process.

Once all the words of the sentences have been evaluated, we have to obtain the emotion for marking up the sentence; the process is different in the case of emotional categories and emotional dimensions:

• Emotional dimensions: Once all the words of the sentences have been evaluated we obtain the average value of each emotional dimension.

• Emotional categories: Once all the words of the sentences have been evaluated we
add up the probability of each emotion of the different words, and we carry out the following process for each of the possible emotions: we process all the emotions in order to obtain the related emotional concepts (the parents of the emotion in the ontology), these related emotional concepts are added to the previous ones with the probability associated to the more specific concept. Then emotions are group by their corresponding level in the ontology and the emotion more general (with the lower level in the ontology) with the bigger probability is assigned to the sentence.

A sample part of a marked tale with emotional categories:

<anxiety>The knight faced the lioness. </anxiety>  
<neutral>He fought she. </neutral>  
<neutral>The knight threw the spear. </neutral>  
...
<delight>She returned to the strong castle. </delight>  
<happy>The knight and the princess lived happy ever afterward. </happy>

And a sample part of a marked tale with emotional dimensions:

<emo val=5.04 act=5.05 cont=5.04>A Fox once saw a Crow fly off with a piece of cheese in its beak and settle on a branch of a tree. </emo>  
<emo val=5.09 act=4.84 cont=4.80>That's for me, as I am a Fox, said Master Reynard, and he walked up to the foot of the tree. </emo>  
...
<emo val=4.52 act=5.03 cont=5.08>In exchange for your cheese I will give you a piece of advice for the future: </emo>  
<emo val=3.41 act=4.85 cont=3.44>Do not trust flatterers. </emo>

5 Evaluation

Four tales took part in the tests. Tales are English popular tales with different number of words (from 153 words and 20 lines to 1404 words and 136 lines). Each of our four tales will be tagged first with the emotional dimensions, and then with the categories.

The data on emotional dimensions we have available for each tale are emotional label for each sentence. For evaluator’s tales we have noticed that the percentage of sentences on which the majority of the human evaluators - half of their number plus one - agrees on the assignment of an emotion is around 70%. This is an important data when it comes to interpreting the results obtained by our tagger. A reference value for the emotion of each phrase is obtained by choosing the emotion most often assigned to that sentence by the human evaluators. In the case of tagger’s tales the reference value obtained in the evaluator’s tales is used.
to compare with the results generated by our tagger. The graph in Figure 4 shows the percentages of success obtained for each tale.

Figure 4. Succes percentage in EmoTag for emotional categories.

6 Future Work

At the moment EmoTag marks up texts taking into account only the lexical emotion, the emotion which derives from the words employed but this is not the only type of emotion that take part in the emotion perceived by the users when they are reading or listening a narration. There are mainly three types of emotions that we have to consider: lexical emotion, semantic emotion and mood of the user.

In order to improve the results we have obtained with EmoTag for the lexical emotions we will consider the following ideas:

- Use a finer granularity for emotional units. We have observed that very long sentences lead to confusion when assigning emotions. For example, we will consider subordinate sentences as emotional units inside a bigger unit, the general sentence.

- Use existing approaches for content analysis of textual data as “The General Inquirer” (Kalin, 1966) which is a dictionary of words marked with different categories; seven of these categories could be interesting for our work: EMOT indicates if the word is related to emotion, POSITIV or NEGATIV marks words of positive or negative outlook, POWER which indicates a concern with power, control or authority or SUBMIT which connotes submission to authority or power or dependence to others, ACTIVE or PASSIVE which marked words as implying an active or passive orientation. Using these tools we can leak the words which are going to take part in our marked up process.

- Processing modifiers. When a modifier appears in a sentence, the emotion associated to that sentence should be increased or reduced.

- Processing modal verbs. When a modal verb appears in a sentence the words under their scope must be treating in a special way. For example, “care to sing” does not imply that the subject is singing so a possible solution could be to reduce the activation because the action is not being held.

- In the first approach of EmoTag only one emotion is attributed to a sentence, the one that is manifested most strongly. In some sentences a better emotional mark up could include not only the strongest emotion but other emotions.

The next step is to get the semantic emotion of a text. We have an ontology of concepts which represents the concepts which take part in a tale and we have ontology of emotions. In order to get the semantic emotion we have to link these two ontologies. This way, we will obtain the emotions related to the concepts that take part in the tale. An example of the relation between concepts (characters and actions) and emotion can be seen in (Francisco et al, 2006). In order to identify the links between a given sentence and the concepts in the domain we can apply text analysis techniques. We have already used techniques such as dependency analysis to identify key concepts of a sentence in order to build conceptual cases from a text corpus (Hervás et al, 2007). This is important, because depending on the semantic content the final emotion could differ from the lexical emotion. For example, the action “to die” is a sadness action, and in our LEW list is marked up this way, but if the subject of this action is the witch the action “to die” turn into a happy action.

Finally, we have the user mood. It seems obvious that the mood of the speaker plays a very important role in determination of the emotional inflection given to an utterance. In order to model this influence we have chosen to consider a representation of the mood of the user as an additional input, modelled along the same lines as the other
emotional information. This can be taken into consideration when it comes to get the final emotion of a text that is going to be present to the user.

Once the three types of emotions are fixed for specific texts it is time to combine them and obtain the final value of emotion for each of the emotional units. We have to study which emotion has the main weight and how each of this type of emotion influence in the others. One way of combine these types of emotions could be, first obtain the lexical emotion, then obtain the emotion associated to the concepts and how the lexical emotion influences in the concepts, and finally modify this emotion with the user mood.

Acknowledgements

This research is funded by the Spanish Ministry of Education and Science (TIN2006-14433-C02-01 project) and a joint research group grant (UCM-CAM-910494) from the Universidad Complutense de Madrid and the Dirección General de Universidades e Investigación of the Comunidad Autónoma de Madrid.

References


Sentiment and near-synonymy: do they go together?

Mary Gardiner
Centre for Language Technology
Macquarie University
gardiner@ics.mq.edu.au

Abstract

Near-synonyms are words that mean approximately the same thing, and which tend to be assigned to the same leaf in ontologies such as WordNet. However choosing between them is still a significant problem for natural language generation systems, as such words may differ in crucial respects, such as in having a positive, neutral or negative attitude to the subject of the generated language; or in having denotational differences that may be important to the discourse.

Previous work in identifying and using near-synonyms has treated these the same and has concluded, on the basis of initial investigations, that a corpus statistics approach is not useful for the problem. However, as they are different, then corpus statistics may still be applicable to a sub-type. In particular if near-synonyms differing in attitude respond better to corpus statistics, this suggests that an approach based on the extensive work in sentiment analysis is worth pursuing. This paper presents initial results showing that this is in fact the case, and presents a research programme based on this.

1 Introduction

The problem of choosing an appropriate word or phrase from among candidate near-synonyms or paraphrases is important for language generation. Barzilay and Lee (2003) cite summarisation and rewriting as among the possible applications, and point out that a component of the system will need to choose among the candidates based on various criteria including length and sophistication. An application of near-synonym generation is the extension of the text generation system HALEogen (Langkilde and Knight, 1998; Langkilde, 2000) to include near-synonyms (Inkpen and Hirst, 2006).

An aspect of the choice between synonyms or paraphrases that should not be neglected is any difference in meaning or attitude. Currently, synonyms and paraphrases are usually treated as completely interchangeable in computational systems. But ideally, for example, a system should be able to make a correct choice between frugal and stingy when trying to describe a person whom the system is intending to praise.

There are several alternative possible approaches to choosing between words like frugal and stingy:

1. choose between them using exactly the same methods as the system uses to choose between fairly semantically unrelated words;
2. choose between them using the same methods that the system uses to choose between any two closely semantically related words, even if they do not differ significantly in attitude (eg battle and fight); or
3. choose between them using a method especially designed for choosing between closely related words with attitude differences.

Some existing work explores using special methods to choose between closely related words (usually near-synonyms) with success better than that of using general natural language generation
techniques to choose between them (Inkpen and Hirst, 2004; Inkpen and Hirst, 2006).

Inkpen et al. (2006) describe techniques for choosing words with either positive or negative sentiment with the aim of producing better texts, but do not specifically justify the need for a special technique to solve the attitude choice problem in particular. In general, especially given the relatively poor performance of the Edmonds (1997) method, the research seems to have tended away from using corpus statistics to solve this problem, at least until the work of Inkpen (2007).

Sentiment analysis work such as that of Pang et al. (2002) and Turney (2002) suggests that it is possible to acquire the sentiment or orientation of pieces of text ranging from words to documents using corpus statistics without needing to use lexicographic resources prepared by experts. This also suggests that the sentiment of a word may affect its collocational context quite broadly. For example, taking two cases from the classification scheme above, it seems intuitively plausible that differences between *placid* (positive) and *unimaginative* (negative) may be expressed throughout the document in which they are found, while for the denotational pair *invasion* and *incursion* there is no reason why the document more broadly should reflect the precise propositional differences that are the essence of the denotational subtype. Therefore, it is possible that the results of the Edmond's experiment vary depending on whether the near-synonyms differ in sentiment expressed towards their subject (attitudinal), or whether they differ in some other way.

In this paper we outline an inquiry into whether approach 2 or approach 3 is more promising: is distinguishing between closely related words that differ in affect different from distinguishing between closely related words that do not differ in affect? In particular, I am exploring whether or not context cues sentiment charged choices more than it cues choices between related words without sentiment differences. Given promising results indicating that context cues may be more important for choosing between sentimentally charged near-synonyms, we outline a possible approach to acquiring such differences automatically.

In Section 2 we outline the general method we are using to test whether sentiment differences between closely related words and other differences between closely related words can be predicted equally well by context or not. In Section 3 we describe an annotation experiment dividing sets of near-synonyms into those differing in attitude and those which do no. In Section 4 we describe results from an early experiment using corpus statistics approaches to discriminate between near-synonyms. In Section 5 we discuss planned future experiments extending the current method and a future research direction incorporating sentiment analysis techniques into acquisition of near-synonym properties.

## 2 Task description

Our test for choosing between closely related words is based on that of Edmonds (1997).

The problem that the system is asked to solve is this: given a set of closely related words, choose which word belongs in a lexical gap in a given sentence. For example, the system might be given the sentence below, with the blank indicating a lexical gap, and asked which of *error*, *mistake* or *oversight* best fits in that gap:

> “However, such a move also of cutting deeply into U.S. economic growth, which is why some economists think it would be a big ____.”

The set of words that the system is asked to choose between might differ in sentiment from each other as, for example *error*, *mistake* and *oversight* do. However, they also might not. For example, the system might be asked to choose between the words *lawyer* and *attorney*, which do not differ in sentiment towards the referent.

The system always uses context cues to choose between the words it is presented with. We compare two sets of test data in terms of how well the system is able to predict the correct word:

1. sets of words where words in each set differ among themselves in affect; and
2. sets of words where words in each set do not differ among themselves in affect.
3 Evaluating near-synonym type

3.1 Method

We conducted an annotation experiment to provide a larger test set of near-synonyms to test our hypothesis against. The annotators were asked to decide whether certain WordNet synsets differed from each other mainly in attitude, or whether they differed in some other way.

The synsets were chosen from among the most frequent synsets found in the 1989 Wall Street Journal corpus. We identified the 300 most frequent WordNet 2.0 (Fellbaum, 1998), where synset frequency is the sum of the frequencies of individual members of the synset. There was no normalisation to compare synsets with different numbers of member words.

Synsets were then manually excluded from this set by the author where they were deemed too similar to other more frequent synsets; were internally similar (for example, the synset consisting of ad, advertisement, advertisement); or contained purely dialectical variation (for example lawyer and attorney). This left 124 synsets of the original 300.

These 124 synsets were then independently annotated by two native English speakers, including the author of this paper, into two distinct sets:
1. synsets that differ primarily in attitude; and
2. synsets that differ primarily in some way other than attitude.

The annotation scheme allowed the annotators to express varying degrees of certainty: they were either definite in their judgement that a synset did or did not differ in attitude; they considered that their judgement was probably correct; or they were completely uncertain.

3.2 Results

Inter-annotator agreement for the annotation experiment is shown in Table 1 both individually for certainty and collectively for all annotations regardless of the annotator’s certainty.

Two divisions of the annotation results were used to compute a κ score and raw inter-annotator agreement: agreement “attitudinal difference”, “not attitudinal difference” and "unsure" regardless of certainty; and agreement between annotators on only the annotations they were definitely sure about, as per Wiebe and Mihalcea (2006). We calculated two κ scores: the Cohen (1960) κ and the Siegel and Castellan (1988) κ; however the two

<table>
<thead>
<tr>
<th>Difference</th>
<th>Certainty</th>
<th>Annotator 1</th>
<th>Annotator 2</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attitude</td>
<td>Definite</td>
<td>14</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Probable</td>
<td>26</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>40</td>
<td>36</td>
<td>29</td>
</tr>
<tr>
<td>Not attitude</td>
<td>Definite</td>
<td>68</td>
<td>63</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>Probable</td>
<td>15</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>83</td>
<td>81</td>
<td>73</td>
</tr>
<tr>
<td>Unsure</td>
<td></td>
<td>1</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Break-down of categories assigned in the annotation experiment

Scores are identical to two significant figures and hence only one κ value is shown.

The results suggest we can be fairly confident in using this classification scheme, particularly restricted to the definite classes.

4 Edmonds' experiment

In the first experiment testing whether near-synonyms differing in attitude are more responsive to corpus statistics techniques than other near-synonyms, we used the methodology of Edmonds (1997). Edmonds' aim was slightly different from ours, in that his work was designed to explore whether contextual cues are sufficient for choosing between closely related words (near-synonyms) in general, rather than to explore whether some sets of closely related words behaved differently from others. However, we can use his method of prediction and then compare the performance of two groups of near-synonyms.

<table>
<thead>
<tr>
<th>Category division</th>
<th>κ score</th>
<th>Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attitudinal, not attitudinal and unable to decide</td>
<td>0.62</td>
<td>82%</td>
</tr>
<tr>
<td>Annotations where both annotators were sure of their annotation</td>
<td>0.85</td>
<td>97%</td>
</tr>
</tbody>
</table>

Table 2: Inter-annotator agreement and κ scores for the annotation experiment
4.1 Method

Edmonds defined a measure designed to determine which of the set of words is cued most strongly by the sentence with the gap, as approximated by similarity score of that word with each of the set of words in the sentence. So the fittingness of, say, error for the gap in the example sentence in Section 2 is approximated by the tendency for each of error and however, error and such and so on to occur together in context.

In general the appropriateness score of any given candidate word \( c \) for sentence \( S \) is the sum of the significance scores \( \text{sig}(c, w) \) for candidate \( c \) with every other word \( w \) in the sentence (barring stopwords):

\[
\text{appropriateness}(c, S) = \sum_{w \in S} \text{sig}(c, w)
\]

The significance score \( \text{sig}(c, w) \) between two individual words is computed as follows (where \( t(a, b) \) is the \( t \)-score for bigrams containing \( a \) and \( b \) in the training data):

1. if the \( t \)-score and mutual information score of \( c \) and \( w \) on the training data are greater than 2.0 and 3.0 respectively, then \( \text{sig}(c, w) \) is given by:
   \[
   \text{sig}(c, w) = t(c, w).
   \]
2. if there is a word \( w_0 \) such that if the \( t \)-score and mutual information score of each of the pairs \( (c, w_0) \) and \( (w_0, w) \) are greater than 2.0 and 3.0 respectively, then \( \text{sig}(c, w) \) is given by:
   \[
   \text{sig}(c, w) = \frac{1}{8} \left( t(c, w_0) + t(w_0, w) \right)
   \]
3. otherwise \( \text{sig}(c, w) = 0 \).

The candidate word \( c \) with the highest score \( \text{appropriateness}(c, S) \) for sentence \( S \) is selected as the chosen word. If there is more than one candidate with that highest score, no candidate is chosen.

The \( t \)-scores and mutual information scores are calculated from bigram frequencies in the 1987 Wall Street Journal using 4 and 10 word windows for bigrams as calculated by the Ngram Statistics Package (Banerjee and Pedersen, 2003).

Edmonds’ method is compared to a baseline, where the most frequent word in any test word set is chosen.

Our test word sets are drawn from the annotation experiment described in Section 3: they are the 58 synsets where the annotators agreed on the type of the synset and were both certain of their judgement. Thus we had 7 word sets agreed to differ internally in attitude and 51 agreed not to.

Our test data consists of test sentences drawn from either the 1987 Wall Street Journal, or the 1988 Wall Street Journal. There were two sets of training data: the 1989 Wall Street Journal using bigrams drawn from 4 word windows around the target word (the experiments called 4win- were trained this way) or from 10 word windows around the target word (the experiments called 10win- were trained this way).

4.2 Results

Since the Edmonds method cannot always make a prediction, we directly compare the baseline and the Edmonds predictions only on sentences where the Edmond method can make a prediction. The number of times that the Edmonds method can make a prediction at all is shown in Table 4, which also shows the baseline correctness on the sentences described, and the Edmonds method correctness where it can make a prediction. A sentence that contains \( n \) words from test synsets is counted as \( n \) separate test sentences in this table.

4.3 Discussion

There are several results of interest here. First, the baselines perform noticeably differently for attitudinal versus non-attitudinal success ratios for each of the five data sets. Calculating the \( z \)-statistic for comparing two proportions, we find that this difference is significant at the 1% level for each of the data sets, with the attitudinal baseline always higher. Similarly, the differences between attitudinal and non-attitudinal success ratios for Edmonds are also significant at the 1% level.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Training window size</th>
<th>Synset size (min 2)</th>
<th>Wall Street Journal year</th>
</tr>
</thead>
<tbody>
<tr>
<td>4win-top3-1987</td>
<td>4</td>
<td>max 3</td>
<td>1987</td>
</tr>
<tr>
<td>4win-top4-1987</td>
<td>4</td>
<td>max 4</td>
<td>1987</td>
</tr>
<tr>
<td>4win-top4-1988</td>
<td>4</td>
<td>max 4</td>
<td>1987</td>
</tr>
<tr>
<td>10win-top3-1987</td>
<td>10</td>
<td>max 3</td>
<td>1987</td>
</tr>
<tr>
<td>10win-top4-1987</td>
<td>10</td>
<td>max 4</td>
<td>1987</td>
</tr>
</tbody>
</table>

Table 3: Test runs for the Edmonds experiment
Because of this first result regarding baselines, the second result, which does show that gross success rates for attitudinal near-synonyms is significantly higher under the Edmonds corpus statistics approach, is less interesting: these higher success ratios could be due to the naturally higher baseline alone.

We inspected some of the data, and noted that for attitudinal synsets, the distribution was much more skewed than for non-attitudinal synsets: one element dominated, and the others were infrequent. In some cases this dominant element appeared to the neutral one, perhaps reflecting the nature of the corpus, but in other cases there was no discernible pattern.

To take into account the varying baselines, we extracted cases where only one method predicted correctly, disregarding those cases where both were right or both wrong. The counts of these are presented in Table 5. We then considered as a 'success' any correct prediction by Edmonds, and calculated the proportion of successes for attitudinal and non-attitudinal for each of the five data sets. Then, for each of the data sets, we compared the success ratios for attitudinal and non-attitudinal, again using the z-statistic for comparing two proportions. The differences are again significant at the 1% level. In this analysis, the attitudinal synsets perform better only for 4win-top3-1987 and 10win-top3-1987; that is, for the cases where there are at most three elements in the synset. For the

<table>
<thead>
<tr>
<th>Test set</th>
<th>Sentences containing test word</th>
<th>Baseline correctness</th>
<th>Edmonds prediction</th>
<th>Edmonds precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attitudinal</td>
<td>Non-attitudinal</td>
<td>Attitudinal</td>
<td>Non-attitudinal</td>
</tr>
<tr>
<td>4win-top3-1987</td>
<td>7588</td>
<td>340246</td>
<td>86.6%</td>
<td>66.2%</td>
</tr>
<tr>
<td>4win-top4-1987</td>
<td>29453</td>
<td>350038</td>
<td>84.0%</td>
<td>65.9%</td>
</tr>
<tr>
<td>4win-top4-1988</td>
<td>27023</td>
<td>295437</td>
<td>85.4%</td>
<td>64.0%</td>
</tr>
<tr>
<td>10win-top3-1987</td>
<td>7588</td>
<td>340246</td>
<td>86.0%</td>
<td>67.8%</td>
</tr>
<tr>
<td>10win-top4-1987</td>
<td>29453</td>
<td>350038</td>
<td>82.7%</td>
<td>67.4%</td>
</tr>
</tbody>
</table>

Table 4: Performance of the baseline and Edmonds method on all test sentences

<table>
<thead>
<tr>
<th>Test data</th>
<th>All words</th>
<th>Attitudinal words</th>
<th>Non-attitudinal words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>Edmonds</td>
<td>Baseline</td>
</tr>
<tr>
<td>4win-top3-1987</td>
<td>9715</td>
<td>10824</td>
<td>5</td>
</tr>
<tr>
<td>4win-top4-1987</td>
<td>13924</td>
<td>12456</td>
<td>604</td>
</tr>
<tr>
<td>4win-top4-1988</td>
<td>11752</td>
<td>10861</td>
<td>594</td>
</tr>
<tr>
<td>10win-top3-1987</td>
<td>28900</td>
<td>20825</td>
<td>14</td>
</tr>
<tr>
<td>10win-top4-1987</td>
<td>37850</td>
<td>23245</td>
<td>1214</td>
</tr>
</tbody>
</table>

Table 5: Number of times each method is right when the baseline and the Edmonds method predict a different word
cases with four elements in the synset, the non-
attitudinal synsets perform better with respect to
the baseline. We speculate that this is due to the
nature of the synsets discussed above: the attitu-
dinal synsets are distributionally very skewed, and
adding a very low probability element (to move
from three to four elements in the synset) does not
make the task of the baseline noticeably harder, but
does add extra noise for Edmonds.

5 Future research plan

The initial experiment provides some support for
the hypothesis that closely related words that differ
in sentiment amongst themselves can be predicted
by context more easily than closely related words
that do not differ in sentiment amongst themselves.
Section 5.1 describes several additional experi-
ments planned that will further test this hypothesis
and Section 5.2 outlines a longer term research
project based on sentiment analysis techniques
aimed at acquiring sentiment differences between
near-synonyms.

5.1 Short term plans

In this section, we outline several further tests of
the original hypothesis that near-synonyms that
differ in attitude are more amenable to corpus sta-
tistics techniques when choosing between can-
didate near-synonyms: Section 5.1.1 describes a me-
thodology for improving the test set; Section 5.1.2
describes a methodology for improving the test
words; and Section 5.1.3 describes using a later
technique which draws on larger amounts of data.

5.1.1 Larger test set

As described in Section 4, our test set consisted of
only 58 word sets, only 7 of which differed in atti-
dude. A stronger version of this experiment, or of
any related experiment, would rely on a larger
number of wordsets. In this section, we describe a
potential source of a larger test set, which could be
used to repeat this first experiment, or as test data
on any of the following experiments.

The General Inquirer wordset (Stone et al.,
1966) is a lexicon of words tagged for various
attributes. In particular, there are 1046 words
tagged as 'Pstv' (positive sentiment) and 1165
tagged as 'Ngtv' (negative sentiment). These words
have been used both in the evaluation of sentiment
analysis systems (Turney and Littman, 2003; Wil-
son et al., 2005) and for use in generation (Inkpen
et al., 2006). Our use of it would be as a source of
test words instead.

There are two possible ways that the General
Inquirer wordset could provide us with a larger test
set:

1. we could select sets of closely related
words that appear in one word list (for ex-
ample, anger and fury, which both appear
in the 'Ngtv' list) as our test set; or
2. we could select sets of closely related
words where there are members of both
the 'Pstv' and 'Ngtv' lists in the set.

Approach 2 is closer to our present experiment,
which concerns the hypothesis that words that dif-
f er in attitude can be predicted more effectively by
their context than other closely related words. Ap-
proach 1 would test a different hypothesis: that
words that merely possess some sentiment in their
meaning, even if all words in the set have the same
polarity of sentiment, can be predicted more effec-
tively by their context. Approach 2 has a disadva-
ntage, however: it is likely to yield a far smaller test
set than approach 1.

5.1.2 Word frequency distributions

Informal inspection of the word sets used for the
experiment in Section 3, together with the good
performance of the most-frequent-word baseline
on the 7 word sets with attitude differences com-
pared to the performance of the baseline on the 51
word sets without attitude differences suggest that
the distribution of the 7 word sets with attitude dif-
f erences tended towards having one highly fre-
quent word together with one or more much less
frequent words, whereas the 51 word sets without
attitude differences tend to have a less dominant
most frequent word.

This tendency may affect the comparison of
their performance in a number of ways:

1. there will be less evidence for the system
to use to choose any of the less frequent
words in the attitude word sets; and
2. the evidence that there is for choosing any
of the less frequent words in the attitude
word sets will be less reliable.

This may affect our ability to directly compare
the attitude word sets with the other word sets.
Therefore, we propose to investigate the distribu-
tion of the word sets in the corpora chosen for fu-
ture experiments more thoroughly. One possible
measure is to compare the entropy (Shannon, 1948) of the relative frequencies of words in the test sets. We can then attempt to choose test word sets with close entropy values.

5.1.3 Inkpen's methodology

Inkpen (2007) describes an alternative approach to the same task as Edmonds (1997) attempted to solve. Instead of estimating the likelihood of a particular near-synonym choice using t-scores acquired from bigrams within a window in the training data, she approximated two mutual information scores from the frequency counts of the words in a one-terabyte corpus as estimated by Clarke and Terra's (2003) Waterloo MultiText System: point-wise mutual information (Church and Hanks, 1991) and PMI-IR (Turney, 2001).

Although she did not test the specific question addressed in this paper of whether or not word sets that differ in attitude are more easily distinguished than those that do not, her overall results suggest that a larger amount of data may produce more coherent results, which may allow for a more effective comparison between the performance of word sets differing in attitude to those which do not. It also reduces the number of relatively arbitrary decisions in the Edmonds method, such as the choice of 2.0 and 3.0 for t-score and mutual information cut-offs, possibly allowing more effective comparisons with other methods.

5.2 Long term plans

Once the question of the differing performance of near-synonyms differing in sentiment is explored, the main thrust of our research will be in applying sentiment analysis approaches to acquiring the differences between near-synonyms and possibly paraphrases.

The goal is to be able to learn, say, the difference between stingy and frugal automatically from unstructured text. A possible methodology could be based on Turney and Littman (2003), where the semantic orientation (positive or negative) of a given word is measured by its association with positive words such as excellent compared to its association with negative words such as nasty, where association can be measured by statistical measures of word association such as Pointwise Mutual Information (Church and Hanks, 1991) or Latent Semantic Analysis (Deerwester et al., 1990). This methodology could be adapted easily to the problem of near synonyms in particular, with the following experimental questions:

1. since near synonyms occur in similar contexts, must the training data be different or larger to reliably distinguish negative words from their positive near-synonyms; and
2. is it possible to use the method to determine “there is no significant polarity difference between these two near-synonyms" as well as determining that one near-synonym is more negative or positive than the other.

If near-synonyms and their sentimental differences can be acquired from free text, we intend to test the effectiveness of using these differences as input to word choice decisions in natural language generation, as Inkpen and Hirst (2006) did with their database of near-synonym differences, acquired from lexicographic resources rather than free text.

References


Abstract

One of the main problems of automatic classification by means of unsupervised machine learning is providing the system with the most adequate training data. In this paper we propose a ‘sentiment zones’ approach for extracting a training subcorpus for sentiment classification. The proposed approach allowed us to increase performance of unsupervised classifiers.

1 Introduction

The present study is a part of a broader research effort aimed at solution of a cross-language sentiment retrieval and extraction (using the Chinese language as a source language and English as a target language). One of the modules of the proposed system is planned to retrieve opinionated texts from Internet. For information retrieval the texts should be indexed with appropriate sentiment tags which in the context of sentiment processing implies classification of the texts according to presence / absence of a sentiment and, if the texts are opinionated, according to their sentiment polarity.

There are two main kinds of machine learning techniques used for classification: supervised and unsupervised. One of the advantages of the unsupervised machine learning is that it does not require expensive manual tagging of training corpus. However, lacking an accurate training corpus, unsupervised learning technique is usually outperformed by supervised machine learning. Another problem that as a rule affects automatic classification (especially sentiment classification) is domain dependency. It is well observed in several papers (see 5) that domain-independent classifiers usually make less correct classifications than ones trained in the same domain. The proposed research addresses the problem of self-adaptation of an unsupervised domain-independent classifier to the corpus. We argue that this can diminish (if not eliminate) the above-mentioned shortcomings of the approach.

2 Data

2.1 Seed Vocabulary

All tests in this study were done using Chinese texts. The Chinese language has some specific features and one of the most distinctive ones is absence of explicit word boundaries, which complicates word-based processing. Although, we consider words to be good units for text processing, we do not regard separate word segmentation as a precondition for NLP tasks in Chinese. For word-based processing we applied the longest-match algorithm to find sentiment words in texts using the NTU sentiment dictionary (NTUSD) (by Ku et al. (2006))⁵. The dictionary has 2809 items in the “positive” part and 8273 items in the “negative”.

2.2 Testing data

The testing corpus was made of product reviews downloaded from web-site IT168⁶. All the reviews

⁵ Ku et al. (2006) automatically generated the dictionary by enlarging an initial manually created seed vocabulary by consulting two thesauri, including tong2yi4ci2ci2lin2 and the Academia Sinica Bilingual Ontological Wordnet
⁶ http://product.it168.com
were tagged by their authors as either positive or negative. Most reviews consist of two or three parts: positive opinion, negative opinion and comments (‘other’), though some reviews have only one part.

After all duplicate reviews were removed the final version of the corpus comprises 29531 reviews of which 23122 are positive (78%) and 6409 are negative (22%). The total number of different products in the corpus is 10631, the number of product categories is 255, and most of the reviewed products are either software products or consumer electronics. Unfortunately some users misused the sentiment tagging facility on the website and quite a lot of reviews were tagged erroneously. However, the parts of the reviews were tagged much more accurately so we used only relevant (negative or positive) review parts as the items of the test corpus. For the tests described in this paper we used 10874 reviews, whose parts were extracted to make a balanced test corpus (5437 items of each sentiment)7.

3 Description of the Approach

We argue that using an automatically extracted subcorpus from a corpus of opinionated texts we can train standard supervised classifiers and achieve results comparable in terms of accuracy to the results of the same classifiers trained on a manually tagged training corpus. We also argue that such a process can be run iteratively to improve the results.

3.1 Sentiment Zone classifier

To extract a training subcorpus we need an unsupervised classifier. Here we propose a Sentiment Zone classifier. The proposed classifier is based on the idea of a sentiment zone. A sentiment zone is a sequence of words or phrases that have same sentiment direction. In this study we use sequence of characters between punctuation marks as a basic sentiment zone. In order to tag a zone as either positive or negative we do the following calculations.

Using Sentiment dictionary (see 2) and maximum match algorithm we extract items (words or phrases) from a zone. As we have two parts of the dictionary (positive and negative), we calculate two scores by formula:

\[ S_{item} = \frac{L_d}{L_{phrase}} \times S_{d} \times N_d \]

where \( L_d \) - length of a dictionary item, \( L_{phrase} \) - length of a phrase in characters; \( S_{d} \) – sentiment score of a word (initially for all the words in the dictionary the score is 1.0); \( N_d \) is a negation check coefficient.

The negation check is a very simple routine based on regex patterns to find out if the word is preceded by a negation within the limits of a phrase. If a negation is found the score is multiplied by -1. Currently we use only five frequent negations: bu, buhui, meiyou, baituo, mianqu, bimian.

The sentiment score of a sentiment zone is the sum of sentiment scores of all the items found in it:

\[ S_{zone} = \sum S_{item(1...n)} \]

Thus we have two alternative sentiment scores for every zone: positive (the sum of all scores of items found in positive part of the dictionary) and negative (sum of the scores for ‘negative’ words). Finally the sentiment direction of a sentiment zone is found by comparing the two alternative scores:

\[ Sentiment_{zone} = \arg\max(S_{c_i}|S_{c_j}) \]

where \( S_{c_i} \) is a sentiment score for one class and \( S_{c_j} \) for the other.

Now we have a number of positive and negative zones in a text. To define the sentiment direction for the whole text, the classifier compares the number of alternative sentiment zones for each item:

\[ Sentiment_{text} = \sum S_{i(1...n)} - \sum S_{j(1...n)} \]

where \( S_{i(1...n)} \) is sentiment zones from 1 to n, that are found in a review. Each zone can be either positive (i) or negative (j). If Sentiment > 0 the whole text is classified as positive and vice versa. If Sentiment = 0 the review is not classified.

In subsequent tests we also use the difference between the number of alternative zones as a thresh-

---

7 The corpus will be made available upon publication of the paper.
old value. The difference is calculated quite similar to the sentiment direction:

\[
\text{Difference} = |\sum \text{Sc}_{i(1,...,n)} - \sum \text{Sc}_{j(1,...,n)}|
\]

As preliminary tests showed, the bigger difference in the number of alternative zones is, the more accurate result we have. The results were evaluated by means of precision\(^8\) and recall\(^9\) (see table 1). The recall of positive reviews was always higher than the one of negatives, thus to get a balanced subcorpus we had to reduce the number of positive reviews. The effective size\(^10\) of extracted subcorpora is indicated in the last column of table 1.

<table>
<thead>
<tr>
<th>Difference</th>
<th>Precision</th>
<th>Recall</th>
<th>Subcorpus size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>87.54</td>
<td>85.45</td>
<td>84.28</td>
</tr>
<tr>
<td>2</td>
<td>95.42</td>
<td>48.65</td>
<td>45.30</td>
</tr>
<tr>
<td>3</td>
<td>97.01</td>
<td>29.93</td>
<td>25.12</td>
</tr>
<tr>
<td>4</td>
<td>97.82</td>
<td>18.18</td>
<td>13.81</td>
</tr>
<tr>
<td>5</td>
<td>98.23</td>
<td>10.96</td>
<td>8.02</td>
</tr>
<tr>
<td>6</td>
<td>98.30</td>
<td>6.53</td>
<td>4.81</td>
</tr>
<tr>
<td>7</td>
<td>97.81</td>
<td>3.78</td>
<td>2.83</td>
</tr>
<tr>
<td>8</td>
<td>98.29</td>
<td>2.16</td>
<td>1.58</td>
</tr>
</tbody>
</table>

Table 1: Influence of the difference between the number of alternative zones on precision and the number of classified reviews

3.2 Retraining

Being able to extract a training subcorpus it is possible to retrain the classifier. The subcorpus is used to adjust the scores of the dictionary items and to find new items (words or phrases) to be included into the dictionary. First of all the subcorpus is pruned of phrases whose relative frequency is very close in different classes. We calculate the frequency difference using following formula:

\[
difference = \frac{|F_i - F_j|}{(F_i + F_j)/2}
\]

where \(F_i\) is relative frequency in one class, and \(F_j\) is relative frequency in another class. Then we look for chunks of texts that have relatively high frequency. Nevertheless duplicate reviews were filtered out from the corpus, there are still a lot of ‘near duplicates’ (reviews with very little difference). To avoid being flooded with parts of such duplicates and to screen out low frequency items we set a threshold for absolute frequency for an item to be more than 5 times per a corpus. For all items, both old ones and newly found, we compare relative frequencies in both classes:

\[
\frac{F_i}{F_i + F_j}
\]

Finally the adjusted dictionary with new scores is ready for new iteration in classifier.

4 Experiments

To test the approaches we designed two experiments. The purpose of the first one is to check if we can get a better training subcorpus after every iteration. A ‘better training subcorpus’ means that at least one of the main characteristics of it (precision or recall) was improved after an iteration while the other one remains almost the same. For example, if we get a higher precision at approximately same recall, the result of the iteration is regarded as positive. The criterion for this improvement is F-measure:

\[
\frac{2PR}{P+R}
\]

, where \(P\) is precision and \(R\) is recall.

The second experiment is to test if the training subcorpora gained by the means of the technique tested in the first experiment can increase accuracy of standard classifiers.

Using the approach that gives us control over the precision and the size of a resulting subcorpus, we want to test two alternative approaches for a training subcorpus extraction: size-driven and precision-driven. The first one means that we use a training subcorpus of maximum possible size, while the latter one means that we use a subcorpus of maximum possible precision providing its size is not too small. It is quite easy to find the threshold value for the size-driven approach: the small-
The greatest difference (1) gives us a subcorpus of 87% of accuracy and of almost 85% of the size of the original corpus. However, we have to use a rather arbitrary notion of a subcorpus for the precision-driven approach. We decided on a subcorpus with 97% of precision and 25% of the size, which is extracted with the difference threshold of 3 (see table 1). As the proposed approach can be used iteratively we will use only these two values throughout all iterations. In the both approaches we use the Sentiment dictionary items as features. We also want to know if modifications made to the dictionary throughout the iterations can also contribute to the performance of the classifiers.

4.1 Experiment 1

In this experiment we ran the Sentiment Zone classifier three times for each difference value. After the first run (see Table 1), we obtained two subcorpora (one for value 1 and the other for value 3) which were used for retraining the same classifier (as described in section 3.2).

4.1.1 Threshold value 1

The results obtained in the run of three iterations for the threshold value 1 are as follows (Iteration 0 is the initial classification, see table 1):

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>87.54</td>
<td>85.46</td>
<td>86.49</td>
</tr>
<tr>
<td>1</td>
<td>90.29</td>
<td>94.29</td>
<td>92.25</td>
</tr>
<tr>
<td>2</td>
<td>90.67</td>
<td>94.49</td>
<td>92.54</td>
</tr>
<tr>
<td>3</td>
<td>90.42</td>
<td>94.74</td>
<td>92.53</td>
</tr>
</tbody>
</table>

Table 2: Results of three iterations (Recall).

From table 2 we can see that iterations 1 - 3 produced a better subcorpus. But as we need a balanced corpus for training, it is important to know if the results are better if we use a notion of corpus size rather then recall.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>87.54</td>
<td>84.28</td>
<td>85.88</td>
</tr>
<tr>
<td>1</td>
<td>87.94</td>
<td>88.64</td>
<td>89.46</td>
</tr>
<tr>
<td>2</td>
<td>86.23</td>
<td>90.47</td>
<td>90.57</td>
</tr>
<tr>
<td>3</td>
<td>88.68</td>
<td>90.91</td>
<td>90.66</td>
</tr>
</tbody>
</table>

Table 3: Results of three iterations (Corpus size in percent to the size of the corpus).

We can see better results after all the iterations.

4.1.2 Threshold value 3

The results of the three iteration for the threshold value 3 are following (Iteration 0 is the initial classification, see table 1): From table 2 we can see that all of the three iterations produced a better subcorpus.

Results for a balanced subcorpus are:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>97.01</td>
<td>29.93</td>
<td>45.75</td>
</tr>
<tr>
<td>1</td>
<td>96.72</td>
<td>41.85</td>
<td>58.42</td>
</tr>
<tr>
<td>2</td>
<td>97.03</td>
<td>45.29</td>
<td>61.76</td>
</tr>
<tr>
<td>3</td>
<td>97.05</td>
<td>45.84</td>
<td>62.27</td>
</tr>
</tbody>
</table>

Table 4: Results of three iterations (Recall).

Judging from table 5 it is possible to conclude that the classifier produced better corpora after all iterations.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>97.01</td>
<td>25.12</td>
<td>39.91</td>
</tr>
<tr>
<td>1</td>
<td>96.37</td>
<td>35.39</td>
<td>51.82</td>
</tr>
<tr>
<td>2</td>
<td>96.43</td>
<td>35.28</td>
<td>51.75</td>
</tr>
<tr>
<td>3</td>
<td>96.24</td>
<td>36.11</td>
<td>52.64</td>
</tr>
</tbody>
</table>

Table 5: Results of three iterations (Corpus size).

4.2 Experiment 2

In this experiment we used the training corpora obtained from the first experiment to train three standard classifiers: a Naive Bayes (NB), a Naive Bayes multinomial (NBm) and a support vector machine (SVM). The baselines (BL) for this experiment are the results performed by the classifiers after being trained using the sentiment dictionary as training corpus. The ‘supervised results’ (SR) are the supervised results obtained from the same set of classifiers, trained on the bigger part (66%) of the test corpus using. These figures are compared with the results of the above named classifiers after they were trained with the subcorpora extracted after each of the four iterations (see tables 3 and 5).

---

11 More iterations did not improve accuracy.

12 We used WEKA 3.4.10 (http://www.cs.waikato.ac.nz/ml/weka)
4.2.1 Original sentiment dictionary items as attributes

In these tests we used the Sentiment dictionary items as attributes for the classifiers. For threshold value 1 the results are:

<table>
<thead>
<tr>
<th></th>
<th>BL</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBm</td>
<td>77.79</td>
<td>81.64</td>
<td>82.11</td>
<td>82.32</td>
<td>82.03</td>
<td>81.85</td>
</tr>
<tr>
<td>NB</td>
<td>50.80</td>
<td>73.74</td>
<td>73.7</td>
<td>73.63</td>
<td>73.49</td>
<td>74.09</td>
</tr>
<tr>
<td>SVM</td>
<td>76.24</td>
<td>80.89</td>
<td>82.63</td>
<td>82.72</td>
<td>82.59</td>
<td>81.77</td>
</tr>
</tbody>
</table>

Table 6: Standard classifiers performance compared (accuracy in percent).

Table 6 shows that all three classifiers increased their performance after being trained on the extracted subcorpus. One of them (Naive Bayes) failed to achieve better results than those gained under the supervised approach. For NB and NBm the best results were achieved with the subcorpus obtained after the second run.

Same settings were used to run the subcorpora obtained after the iterations with threshold value 3:

<table>
<thead>
<tr>
<th></th>
<th>BL</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBm</td>
<td>77.79</td>
<td>80.99</td>
<td>81.01</td>
<td>80.95</td>
<td>81.11</td>
<td>81.85</td>
</tr>
<tr>
<td>NB</td>
<td>50.80</td>
<td>74.2</td>
<td>73.11</td>
<td>72.73</td>
<td>72.73</td>
<td>74.09</td>
</tr>
<tr>
<td>SVM</td>
<td>76.24</td>
<td>80.77</td>
<td>81.65</td>
<td>81.89</td>
<td>81.79</td>
<td>81.77</td>
</tr>
</tbody>
</table>

Table 7: Standard classifiers performance compared (accuracy in percent).

The results presented in this table are slightly worse than those in table 6, all classifiers outperformed the baseline, and only NBm failed to produce better results than a supervised classifier.

4.2.2 Modified sentiment dictionary as attributes

In these tests we used modified Sentiment dictionary items (as explained in section 3.2) as attributes for the classifiers. Thus, the classifiers were not only trained on extracted subcorpora, but also used different set of attributes for each run. For threshold value 1 the results are:

<table>
<thead>
<tr>
<th></th>
<th>BL</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBm</td>
<td>77.79</td>
<td>88.18</td>
<td>88.32</td>
<td>88.54</td>
<td>88.13</td>
<td>81.85</td>
</tr>
<tr>
<td>NB</td>
<td>50.80</td>
<td>82.28</td>
<td>81.57</td>
<td>80.86</td>
<td>81.65</td>
<td>74.09</td>
</tr>
<tr>
<td>SVM</td>
<td>76.24</td>
<td>84.64</td>
<td>87.9</td>
<td>88.19</td>
<td>88.00</td>
<td>81.77</td>
</tr>
</tbody>
</table>

Table 8: Standard classifiers performance compared (accuracy in percent).

Table 8 shows that all three classifiers outperformed not only the baseline but also the supervised results. It is interesting to note that after all iterations the accuracy was higher than the one of supervised classifier.

Same settings were used to run the subcorpora obtained after the iterations with threshold value 3:

<table>
<thead>
<tr>
<th></th>
<th>BL</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBm</td>
<td>77.79</td>
<td>84.55</td>
<td>85.68</td>
<td>85.41</td>
<td>85.41</td>
<td>81.85</td>
</tr>
<tr>
<td>NB</td>
<td>50.80</td>
<td>77.8</td>
<td>81.3</td>
<td>80.94</td>
<td>81.09</td>
<td>74.09</td>
</tr>
<tr>
<td>SVM</td>
<td>76.24</td>
<td>82.72</td>
<td>84.53</td>
<td>85.7</td>
<td>85.52</td>
<td>81.77</td>
</tr>
</tbody>
</table>

Table 9: Standard classifiers performance compared (accuracy in percent).

The results presented in this table are again worse than those of threshold value 1 (see table 6), but all of the classifiers outperformed the supervised approach, and, again, after all iterations.

5 Related Work

The approach presented in this paper is very close to the bootstrapping technique described by Yarovsky (1995): we use automatically built Sentiment dictionary to extract a training subcorpus to train a classifier. The process can be run iteratively to increase precision and coverage.

Sentiment classification using supervised machine learning was studied by Pang et al. (2002). The authors showed that machine learning methods (Naive Bayes, maximum entropy classification, and support vector machines) with words as features do not perform as well on sentiment classification as on traditional topic-based categorization: the best accuracy achieved was 82.9%, using an SVM trained on unigram features. A later study Pang and Lee (2004) increased performance up to 87.2%, but the object of classification was an opinionated sentence, not a text (review or its part). Pang et al. also showed that bigrams are not effective at capturing context in sentiment extraction, but modelling the potentially important contextual effect of negation had some positive influence on performance. Following these findings we also used a unigram-presence approach as well as implemented a simple negation check. The main difference is that the training corpus we used is automatically generated, this enables us to regard our
Das and Chen (2006) designed an algorithm which comprises different supervised classifier algorithms coupled together by a voting scheme for extracting small investor sentiment from stock message boards. Among the others they use a classifier algorithm based on a word count of positive and negative connotation words. It makes this study close to ours as we also use words counts for calculation sentiment scores. The difference is that the dictionary we use (Sentiment Dictionary by Ku et al., see 2) was generated automatically, rather than manually crafted. The same (manual) approach to sentiment lexicon construction combined with fuzzy-logic was used by Huettner and Subasic (2001).

Hu and Liu (2004), Kim and Hovy (2004), Ku et al. (2006) create a sentiment dictionary by means of a set of seed words which is enlarged using other dictionaries or thesauri, and the automatically generated dictionaries are then used for classification. We use such a dictionary iteratively 1. to adjust it to the corpus and 2. to obtain a training subcorpus for machine learning classifiers.

Turney (2002) proposed an unsupervised learning algorithm for classifying a review where the sentiment direction of a phrase is calculated as the pointwise mutual information (PMI) between the given phrase and the word ‘excellent’ minus the PMI between the given phrase and the word ‘poor’. The author reports different accuracies (from 64% to 84%) obtained after evaluation of classifications in different domains, with average accuracy of 74% on 410 reviews. Although we do not use a PMI for calculating semantic orientation, we use sentiment scores of words for calculating sentiment orientation of a phrase in a very similar way: by comparing negative and positive scores.

Aue and Gamon (2005), Eriksson (2006) and Read (2005) amongst others have noted the influence of topic- and domain-dependency in sentiment classification; the authors observed that a classifier performs much better if trained on data from the same domain as testing data.

Liu et al (2005) use notion of opinion segment and of opinion set of a feature, which are close to the notion of sentiment zone used in this paper. Both concepts denote a chunk of syntactic units that are characterized by some sentiment direction. The difference is that the first two are one or more sentences long whereas we operate with phrases. Another difference is that Liu et al. use these chunks for analyzing and comparing opinions regarding a product feature, thus they are ‘product feature-driven’: they are located after a feature is found.

6 Conclusion and future work

The experimental data shows that the proposed approach helps to increase performance of an unsupervised sentiment classifier and outperform the supervised approach. It was also observed that in these tests a bigger size of a training corpus was more important rather than higher precision to achieve better accuracy. A major improvement of accuracy was achieved by using a modified set of attributes combined with an extracted training subcorpus. The proposed approach may also be domain-independent as it uses a domain-independent dictionary in the first iteration, but it needs validation by testing it on different corpora. Another question to be answered is why the increase in performance was not linear.

One of the problems to be solved is the problem of a very unbalanced output of the sentiment zone classifier: although the approach seems to be efficient for both negative and positive items, their performance is too different. Positive reviews increase precision keeping relatively high recall, while precision of positive reviews is accompanied by a rapid loosing of recall. This phenomenon forces us to make the corpus smaller in order to avoid complications of processing a skewed corpus. The disbalance may have been caused either by the classifier while the sentiment score calculation or it may be a distinct feature of negative reviews in general: people tend to be less emotional and more reasonable. It means that we have to seek for ways of improvement the scoring technique as well as to try some linguistic features that might contribute to more accurate scores. First of all we would like to test the negation check again as at present this routine is implemented in a rather simplistic way. Another feature to be tested is sentiment intensifiers (words such as “very”, “absolutely” and others). It also may be beneficial to increase the score for subjectivity indicators (word combinations like “I think” and some interjections). Probably a more sophisticated technique may be applied to filtering and processing the training subcorpora. So far we
use quite a simple technique based on comparison of relative scores. The positive influence of the modified set of attributes on the accuracy of classification enables us to pay special attention to this phenomenon and investigate better ways of extracting and modifying attributes. All the more, this domain (words and phrases extraction) is of special interest in the context of Chinese linguistics, where the problem of the word definition is far from being solved, which significantly affects Chinese NLP.

References


Sanjiv R. Das and Mike Y. Chen. 2006. Yahoo! For amazon: Sentiment extraction from small talk on the web.


2 Models of Language
A mereology-based general linearization model for surface realization

Ciprian-Virgil Gerstenberger  
Computational Linguistics  
University of Saarland, Germany  
gerstenb@coli.uni-sb.de

Abstract

In this paper, we propose a cross-linguistically motivated architecture for surface realization based on mereology, i.e. on the part-whole distinction. We describe the natural language utterance in terms of embedded Linear Order Parts with two mutually exclusive relations holding between them: Part-Of relation and Linear Order relation. We use the mereological utterance description to design a general linearization model which we motivate for the surface realization architecture we propose: a linearization module, an inflection module, and a text polishing module. This architecture models surface realization phenomena in terms of constraints on grammatically valid configurations of utterance “parts”.

1 Introduction

One of the challenges for multilingual Natural Language Generation (NLG) systems is to provide flexible, context-sensitive output, taking into account the mode of interaction: spoken vs. written. In particular, the surface realization's dependence on context is manifested in the utterance's word order.

In order to account for context-dependent word order phenomena, various linearization models and formalisms have been proposed (Habash et al., 2001; Bohnet, 2004; Gerdes and Kahane, 2001). While the approaches are mainly language-independent, basic questions such as what are appropriate candidates for primitive items for language-independent linearization or what are the subsequent steps of linearization that lead to orthographically valid strings, are left open.

We propose a cross-linguistically motivated General Linearization Model (GLM) that refers to a linguistically informed Mereological Utterance Description (MUD), and addresses the questions above. Using examples from three languages - Polish, German, and Romanian -, we show that this model offers a solid ground for a uniform treatment of various linguistic phenomena related to linearization.

This paper is organized as follows: In Section 2, we present basic observations that motivate a language-independent surface realization. In Section 3, we propose the MUD as a fundament for a flexible, language-independent linearization. Section 4 deals with essential questions for a general surface realization. In Section 5, we present the GLM, and briefly describe the surface realization architecture. We exemplify linearization with GLM in Section 6, discuss related work in Section 7, and summarize our contribution in Section 8.

2 Motivation

In working towards a language-independent surface realization model, we confronted the problem of finding the most general features of natural language that are relevant to our task. The model we arrived at is motivated by the following observations:

Observation 1 Speech is prior to writing both culturally and historically (Greenberg, 1968).

Given this observation, phonological knowledge may be used for proper orthographic surface realization (but not vice-versa).
Observation 2 Generation and analysis are two fundamentally different tasks.

The input for analysis is a string. The process of syntactic analysis may make use of empty elements (traces, empty topological fields\(^{13}\), etc.) to build complete syntactic structures. For linearization, there is no need for empty elements.

Observation 3 The smallest linearizable entity in a language is the phoneme.

Various types of speech errors - phoneme shifts (e.g.,\(\text{mutlimodal}\)), phoneme cluster shifts (e.g.,\(\text{flow snurries}\)) or morpheme shifts (e.g.,\(\text{self-instruct destruction}\)) - show that units smaller than words are subject to linearization.

Observation 4 The most general relation between two entities \(\alpha\) and \(\beta\) such that \(\alpha\) is a substructure of \(\beta\) is the part-of relation.

A phoneme is part of the syllable containing it. This also holds for a word and the constituent containing that word or for a constituent and the clause containing that constituent. However, what is regarded as a constituent depends on which constituency tests are used, and traditional constituency\(^{14}\) tests are controversial (Phillips, 2003; Miller, 1992). By contrast, there is no controversy about - and no need to test - the fact that the phoneme \(/n/\) is part of the syllable \(/no/\).

Note that the “part-of” concept is an old philosophical concept, and that regarding language entities as part-of structures is not a novelty either (Smith, 2001; Moravcsik, to appear).

A linguistic theory that employs the part-of concept for cross-linguistic analysis is Radical Construction Grammar (Croft, 2001). Croft says that “[...] the only syntactic structure in constructions is the part-whole relation between the construction and its elements.” (Croft, 2001)

Based on the observations above, we search for a cross-linguistic utterance description level.

3 Mereological Utterance Description

Taking into account the observations in the previous section, we propose a mereological utterance description. First, we define the unit of description, and then, we present relations and properties of mereological structures.

Definition 1 A Linear Order Part (LOP) is a language item which is phonologically realized as a contiguous part of a grammatically well-formed utterance.

According to the definition above, following linguistic entities are LOPs: a phoneme (e.g., \textit{cluster}), a phoneme cluster - not necessarily a syllable (e.g., \textit{cluster}), a syllable - not necessarily a morpheme - (e.g., \textit{cluster}), a morpheme - not necessarily a free morpheme – (e.g., \textit{incredible}), a word (e.g., \textit{a book}), parts of adjacent words (e.g., \textit{a big red book}), or word groups (e.g., \textit{the rather boring book}). Hence, any contiguous part of a grammatically well-formed utterance is a LOP. A LOP can be either motivated linguistically - e.g., (a) contiguous constituents such as noun phrases, adjectival phrases, (b) partial constituents, (c) non-empty topological fields, (d) embedded clauses, etc. - or not motivated linguistically - e.g., \textit{the nice book}. We restrict the use of MUD to LOPs that are linguistically motivated\(^{15}\).

The following two relations hold between LOPs:

Definition 1 [Part-Of (PO) relation]
Let \(\lambda_1\) and \(\lambda_2\) be two different LOPs: \(\lambda_1 \subseteq \lambda_2\) iff \(\lambda_1\) is proper part of \(\lambda_2\). The PO-relation is reflexive, antisymmetric, and transitive.

Definition 2 [Linear Order (LO) relation]
Let \(\lambda_1\) and \(\lambda_2\) be two different LOPs: \(\lambda_1 \prec \lambda_2\) iff the occurrence of \(\lambda_1\) precedes the occurrence of \(\lambda_2\) in the utterance. The LO-relation is irreflexive, asymmetric, and transitive.

In addition, LO-relation and PO-relation are mutually exclusive, i.e., two different LOPs can either PO-relate or LO-relate, but not both.

Definition 4 [Exclusivity]
Let \(\lambda_1\) and \(\lambda_2\) be two different LOPs, then:

1. if \(\lambda_1 \subseteq \lambda_2\), then \(\lambda_1 \not\prec \lambda_2\) and \(\lambda_2 \not\subseteq \lambda_1\)

\(^{13}\)Topological Field Model (TFM) for German (Höhle, 1983)

\(^{14}\)There are syntactic theories that use “flexible”, non-traditional constituents, e.g., Combinatory Categorial Grammar (Steedman, 2000).

\(^{15}\)It is obvious that MUD can cover all types of speech errors, but this is not part of our task.
4.1 Inflected or non-inflected items?

There are two opinions regarding the order of syntax-morphology processing: syntax comes before inflection morphology (e.g., Government and Binding (Chomsky, 1981/1993), Minimalist Program (Chomsky, 1995); syntax comes after inflection morphology (e.g., Lexical Functional Grammar (Bresnan, 2001), Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994)).

Let us consider an example. In the Romanian this-NP, the position of the demonstrative can be either prenominal (acest om, this man) or postnominal (omul acesta, this man) (Mallison, 1986; Bărbuță et al., 2000; Constantinescu-Dobridor, 2001).

<table>
<thead>
<tr>
<th>Prenominal</th>
<th>Postnominal</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) acest om this man</td>
<td>(5) omul acesta man this</td>
</tr>
<tr>
<td>(2) *acest omul this this</td>
<td>(6) *omul acest this</td>
</tr>
<tr>
<td>(3) *acest om this det man</td>
<td>(7) *omul acest man this</td>
</tr>
<tr>
<td>(4) *acest omul this det man</td>
<td>(8) *omul acest man this</td>
</tr>
</tbody>
</table>

The Romanian this-NP is always definite, but it shows different marking patterns depending on the relative position of the demonstrative with respect to the noun. In prenominal position, neither the demonstrative nor the noun is marked for definiteness (1), while in postnominal position, both the demonstrative and the noun is marked for definiteness (5). All other possible combinations are not grammatically correct (2)-(4), (6)-(8).

To obtain only the two grammatically correct variants of the Romanian this-NP, (1) and (5), both the morpho-syntactic specification and the relative position of the demonstrative with respect to the noun are required. This fact definitely speaks for linearization before inflection morphology. The conclusion is that theoretical frameworks such as HPSG or LFG might not be able to generate all grammatically correct variants of a Romanian this-NP without explicit coding of linearization-relevant information in other processing modules that are not supposed to handle linearization.

4.1.2 Lexical or sublexical items?

It is not always possible to tell whether an item is an affix, a clitic, or a word (Miller, 1992). This is understandable, given that language change is an ongoing process: “[l]anguage is fundamentally
Synchronic language states are just snapshots of a dynamic process emerging originally from language use in conversational interaction.” (Croft, 2001)

To illustrate linearization-relevant sublexical phenomena, let us consider the Polish person-number marker (PN-marker) in past tense.

(9) Nie widzieliśmy tego. [We didn’t see this.]
not see-pst-m-pl-1pl this

(10) Tegośmy nie widzieli
this-1pl not see-pst-m-pl

(11) Myślimy tego nie widzieli
we-1pl this not see-pst-m-pl

While preverbally the PN-marker behaves like a clitic attaching to various words (10)-(11), post-verbally it behaves like a suffix, attaching only to the finite verb (9), hence the term floating affix (Kupść and Tseng, 2005; Crysmann, 2006).

To find out the granularity of primitive linearization entities, we propose the following test.

**Linearization test** Given two items α and β at morpho-syntactic level in a specific language: if the language allows for both α < β and β < α, then these items are linearization primitives.

Let us illustrate the application of the test to the following cases: the Polish PN-marker, the Romanian weak pronoun, and the German separable verb particle.

(12) Säi facei! [Do it]
that it do-conj-2pl

(13) Säi facei!
that it do-conj-2pl

(14) Facei-l!
do-imp-pl it

(15) Sie will das Fenster aufmachen,
she wants the window off make
She wants to open the window.

(16) Sie macht das Fenster auf,
she makes the window off
She opens the window.

The Polish PN-marker can occur before (10)-(11) or after the verb (9). The Romanian weak pronoun can occur before (12)-(13) or after the verb (14). Finally, the German separable verb particle can occur before (15) or after the verb (16).

Taking into account the phenomena described above, we propose that the set of primitive items for linearization should contain both sublexical items that pass the linearization test and lexical items, provided that there is agreement among linguists about the definition of the lexeme.

### 4.2 How to form complex items?

According to the MUD, the utterance is a structure of embedded LOPs. A complex LOP is a LOP consisting of at least two LOPs. Assuming that, for linearization, the primitive LOPs described in the previous section are simple LOPs, we show in this section how to form complex LOPs.

Given the well-known phenomena of discontinuous constituents such as partial fronting and extraposition, it is obvious that forming complex LOPs does not necessarily correspond to forming (traditional) constituents. If a language allows complex LOPs to occur in different positions, a general mechanism of forming complex LOPs should take the following constraints into account:

1. whether two or more primitive LOPs permute **always** as a unit;
2. whether two or more primitive LOPs permute **sometimes** as a unit, and if so, under what circumstances.

We call the first Total Permutation Constraint (TPC) and the second Partial Permutation Constraint (PPC). If, in a specific language, two or more primitive LOPs never permute as a unit, no complex LOP can be formed of them.

To illustrate the TPC, let us consider the German *article+noun* combination (17)-(19). In German, the article and the noun permute as a unit, independent of their occurrence in a grammatically correct utterance.

(17) Das Buch ist schön.
the book is nice
The book is nice.

(18) Schön ist das Buch.
nice is the book
The book is nice.

(19) Ist das Buch schön?
is the book nice
Is the book nice?

Now imagine an - admittedly strange - language in which the article of the direct object of a verb, the subject, and the temporal adverbial can per-
mute freely, but always as a unit. They do not form a constituent, but for linearization they are a unit (like the German article+noun combination).

It is clear that while contiguous constituents always meet TPC, TPC applies to all kinds of LOP combinations, which are not necessarily semantically related. The strange language example above illustrates explicitly this important issue.

Note that the fact that complex LOPs in our model are built solely based on TPC/PPC, and not on - more or less traditional - syntactic constituency, is one of the crucial differences between the model we propose and approaches that might at first sight resemble it, such as (Bohnet, 2004; Gerdes and Kahane, 2001; Steedman, 2000; Bozsohin, 2002).

We want to point out that TPC/PPC and adjacency are not the same, and that adjacency is not an appropriate linearization constraint: if there are only two primitive LOPs to combine, adjacency is automatically imposed by TPC/PPC. However, imagine for instance the German das rote Buch (the red book): these primitive LOPs meet TPC as well - they always permute as a unit -, but, in this constellation, the article is never adjacent to the noun. Moreover, just putting two or more LOPs together does not say anything about their position with respect to each other, as is the case with scrambling in the middle field in German.

To illustrate the PPC, let us consider German extraposed and non-extraposed relative clauses such as in Peter hat gestern ein Buch, das schön ist, gekauft (Yesterday, Peter bought a nice book).

(20) Peter hat gestern ein Buch, das schön ist, gekauft. (Peter has yesterday a book that is nice bought)

(21) Peter hat ein Buch, das schön ist, gestern gekauft. (Peter has a book that is nice yesterday bought)

(22) *Peter hat ein Buch gestern, das schön ist, gekauft. (Peter has yesterday a book that is nice bought)

(23) Peter hat gestern ein Buch gekauft, das schön ist. (Peter has yesterday a book bought that is nice)

(24) Peter hat ein Buch gestern gekauft, das schön ist. (Peter has yesterday a book bought that is nice)

As long as both ein Buch and das schön ist occur in the middle field (the underlined part in the examples above), they have to form a complex LOP that can be scrambled as a whole (20)-(22). However, if the relative clause is extraposed, the adverb gestern can occur between the noun and the relative clause (23)-(24). In the same vein, we model linearization constraints stemming from different description levels: morpho-syntactic, syntactic and macro-structural.

4.3 How to linearize?

The linearization process can be performed in two ways: either using absolute positions (“Put item α in the nᵗʰ, i.e., in a predefined, position!”) or using relative positions (“Put item α before item β?”) for linearization items (Ahrenberg, 1999). Since we assume that all linearization-relevant items - both content and function items - are already provided by the previous NLG steps, we propose a linearization expressed in terms of before (or after).

5 General Linearization Model

In this section we introduce the General Linearization Model we arrived at using MUD. First, we describe the linearization entities, the input structure for linearization, and the rules that constrain the linearization process. Then, we sketch a surface realization architecture based on GLM.

5.1 Linearization entities

The linearization entities of the GLM reflect the LOPs of the MUD in that they are abstractions of the (concrete) LOPs (see Definition 1).

Definition 6 A Symbolic Linear Order Part (SLOP) is a symbolic representation of a language item which has to be phonologically (or graphically) realized as a contiguous part of a grammatically well-formed utterance (or sentence).

Primitive SLOPs are symbols for linearization-relevant items at morpho-syntactic level, i.e., content words, function words, and sublexical items that pass the linearization test. Each SLOP contains its morpho-syntactic specification. For instance, the specification for the German form machte is [MACHEN, type=verb, vForm=fin, temp=imperf, pers=3, nr=sg], the specification for the Romanian weak pronoun forms îl and -l is [PRON, type=weak, pers=3, nr=sg, gender=n, case=acc], and the specification for the Polish PN-marker form śmy is [PN-MARKER, pers=2, nr=pl].
5.2 Input structure
The input structure for linearization is a dependency tree whose nodes are primitive SLOPs. This means that the module responsible for building input structures has to know about primitive linearization items. As with other dependency models (Gerdes and Kahane, 2001; Bohnet, 2004), we assume that there is no explicit linearization information stored with the tree.

From the ID/LP grammar perspective (Shieber, 1984), the task can be described as follows: given the ID structures, find all possible LP variants and rank them according to their appropriateness to the specific communicative situation.

5.3 Rules
Reflecting MUD, GLM features two types of rules: PO-relating rules (mereological rules) and LO-relating rules (linear order rules).

5.3.1 Part-Of rules
The PO-rules constrain the formation of complex SLOPs according to the total and partial permutation constraints of a given language (see Section 4.2). For German, such a rule forms a complex SLOP from every subtree of the form \([\text{noun} – \text{det} \rightarrow \text{art}]\), constraining the permutability of a noun and its article (see (17)-(19)).

5.3.2 Linear Order rules
The GLM features three types of LO-relating rules (see Section 4.3):

1. **vertical**: constraining the position of a node with respect to its children nodes
   
   In German, for every subtree of the form \([\text{noun} – \text{det} \rightarrow \text{art}]\), the SLOP for article precedes the SLOP for noun.

2. **horizontal**: constraining the position of a node with respect to its sibling nodes
   
   In German, for every subtree of the form \([\text{noun} – \text{det} \rightarrow \text{art}] \& [\text{noun} – \text{mod} \rightarrow \text{adj}]\), the SLOP for article precedes the SLOP for adjective.

3. **diagonal**: constraining the position of a node with respect to nodes that are neither siblings nor in immediate dominance relationship in the dependency tree

   This rule constrains the position of a node or a subtree which does not form a complex SLOP with its mother SLOP, as is the case with the German extraposed relative clauses (see (23)-(24)).

5.4 Surface realization architecture
Based on GLM, we propose the following surface realization architecture:

1. a **linearization module** that takes as input dependency trees as described above and builds all valid variants according to the PO- and LO-rules of a given linearization grammar. The output of the linearization step is a set of projective SLOP-trees.

2. an **inflection module** that takes as input the output of linearization and, based on the morpho-syntactic specification and, if needed, on positional information, obtains the appropriate surface forms of the individual SLOPs plus their phonological representation. The output is the set of the same projective SLOP-trees enriched with morpho-phonological representations.

3. a **text polishing module** that takes the output of the previous step and, based on the surface form and the context of each SLOP, performs phonological assimilation, orthographic editing, and punctuation. The output of this module is the set of the final surface realization variants of the utterance.

6 Examples
This section exemplifies the linearization process with the architecture described above.

**Polish** Let us consider the Polish PN-marker as in (9)-(11). For the given input, the PO-rules form complex SLOPs (see Figure 1). The LO-rules for Polish order simple and complex SLOPs. After the inflection morphology step, the string is processed as follows: the first word form is capitalized; the
PN-marker is joined to the preceding word, if no phonological constraint is violated as a result, otherwise the variant is rejected (Kupść and Tseng, 2005); then, punctuation rules are applied (see Figure 2).

For Romanian weak pronouns (12)-(14), the result of linearization and inflection is să îl faceți and faceți îl. Based on the optional weak pronoun enclitization to the subjunction să in preverbal position, the text polishing module generates both Să îl faceți? and Să-îl faceți! from the first variant; based on the obligatory weak pronoun enclitization to the verb in postverbal position, only Faceți-îl! is generated from the second variant.

7 Related work

The most important concept of GLM is the use of mereological structures as “generalized constituents”. Dissociating linear order from constituency is not new (Reape, 1994; Pollard et al., 1994; Kathol, 1995; Goetz and Penn, 1997), but the proposed models are concerned mainly with analysis, and from the generation perspective, especially regarding surface realization, many questions are left open.

For generation, (Gerdes and Kahane, 2001) propose a Topological Dependency Grammar to model syntactic and macro-structural phenomena in German, but this model is limited to word-level phenomena. Thus, linearization-relevant sublexical phenomena are not taken into account.

Regarding linearization-relevant sublexical phenomena, the same applies to the model proposed in (Bohnet, 2004). Bohnet's model is similar to ours in terms of the input structures and ways of expressing linearization rules. However, there are basic differences with respect to both primitive and complex linearization items. While our model employs mereological-based units that abide by TPC/PPC (see Section 4.2), Bohnet's precedence units “roughly represent constituent structures.” (Bohnet, 2004)

Moreover, Bohnet's model uses only two kinds of LO-relating rules - vertical and horizontal -, failing possibly to constrain extraposed relative clauses in German properly.

Unlike (Minnen et al., 2000), the GLM-based surface realization architecture we propose allows for different variants at the level of morpho-syntax, as exemplified by means of Romanian weak pronouns in the previous section.

**Romanian** For the Romanian this-NP, the LO-rules impose no restriction on the position of the demonstrative with respect to the noun: both variants, (1) and (5), are possible. The result of linearization is both DEM < OM and OM < DEM. Using their relative positions with respect to each other, the underspecified feature values for definiteness for both demonstrative and noun can be computed. Then, all valid variants can be realized: acest om for the constellation [DEM, def=−] < [OM, def=−], and omul acesta for the constellation [OM, def=+] < [DEM, def=+].
8 Conclusions

In this paper, we presented a mereological utterance description, a general linearization model that draws upon MUD, and a surface realization architecture that draws upon GLM. The overall goal of this undertaking is a flexible, language-independent model for surface realization. We asked basic questions about linearization, and tried to find answers to them by addressing relevant, cross-linguistic phenomena.

In the future, we intend to implement the theoretical ideas presented here. Our research topic is to identify, classify, and implement in a modular manner linearization constraints specific (not only) to the languages discussed in this paper. The ultimate goal is to provide appropriate ranking of linearization variants with respect to the specific communicative situation.

References


M. Reape. 1994. Domain union and word order variation in German. In German in HPSG, Stanford University.


Improving target language modeling techniques for statistical machine translation

Maxim Khalilov
Departament de Teoria del Senyal i Comunicacions
Centre de Recerca TALP (UPC)
Barcelona 08034, Spain
khalilov@gps.tsc.upc.edu

Abstract

The aim of this study is to find ways of improving target language modeling (TLM) applied to statistical machine translation (SMT). We describe current research activities dedicated to TLM improvement that are applied to the 2007 n-gram-based statistical machine translation system developed in the TALP Research Center at the Technical University of Catalonia (UPC).

We consider two new language modeling improvement techniques: threshold-based TLM pruning and TLM based on statistical classes. Some of the research is still in progress. In this paper we describe some of the major problems faced and outline possible solutions and plans for future research.

We describe the results for the Spanish-English and English-Spanish language pairs from the official TC-STAR 18 2006 evaluation.

1 Introduction

The statistical approach to machine translation is not a new area of scientific research and was actually one of the earliest fields in computer science. SMT lost popularity during the period from 1960 to 1980 but interest was renewed in the early 1990s and has grown rapidly in recent years due to the potential of the approach.

One of the major advances in the accuracy of translation systems was achieved by changing from early systems based on the noisy channel model, which performed word-to-word translation (Brown et al., 1993), to phrase-based systems, which are based on the same approach and use aligned bilingual corpora to translate bilingual units (Koehn et al., 2003; Zens et al., 2002).

The n-gram-based approach appeared during the same period (Mariño et al., 2006). The main difference between the two modern approaches can be found in the representation of bilingual units defined by word alignment (Crego et al., 2005a).

Language modeling is widely used in a large number of human language technology applications, including SMT. It can either be an integrated component or an additional feature depending on the approach on which the translation system is built. However, it significantly affects the system performance in both cases.

It is well known that LMs can often be very large and sometimes cause memory overflow problems. LM pruning is definitely required, but it reveals an efficiency-performance trade-off which generally causes decreased performance in smaller models. However, carefully determined pruning can reduce system noise and increase translation quality. In this study we consider a possible LM pruning strategy based on rational threshold selection.

We also focus on the use of the word class TLM as a feature of the log-linear model. The system introduces two types of word classes: statistical and linguistic.

The rest of the paper is organized as follows. In Section 2 we briefly outline the UPC n-gram translation system, system models, decoding and opti-

---

18 TC-STAR (Technology and Corpora for Speech to Speech Translation). The project web-site is http://www.tc-star.org
mization procedures from 2006. In Section 3 we describe the framework of the TC-STAR 2006 evaluation (further details can be found on the ELDA web-site: www.elda.org/en/proj/tcstar-wp4). Section 4 contains the experimental results and in Section 5 we present the conclusions and explain plans for future research.

2 Statistical machine translation

SMT is based on the principle of translating a source sentence $f$ (traditionally French) into a sentence in the target language $e$ (English). The problem is formulated in terms of source and target languages and is defined as an $arg\ max$ operation according to the following equation:

$$
\hat{e}' = \arg\ max_{e'} \left\{ p(e' | f') \right\}
$$

(1)

where $I$ and $J$ represent number of words of the sentences in the target and source languages, respectively. Consequently, the translation problem can be reformulated as selecting a translation with the highest probability from among a set of target sentences.

Modern SMT developed from the work carried out by IBM in the early 1990s. This research was largely inspired by experiments in the field of speech recognition. The first SMT systems were based on the noisy-channel approach (Brown et al., 1990) and performed word-level translation. The previous equation can be decomposed according to the Bayes rule as follows:

$$
\hat{e}' = \arg\ max_{e'} \frac{p(f' | e') \cdot p(e')}{p(f')}
$$

(2)

Therefore, the problem of finding conditional probability becomes an $arg\ max$ operation of the product of two models:

- $P(f|e)$ refers to a bilingual translation model (BTM) probability.
- $P(e)$ refers to a target language model (TLM) probability.

Modern SMT models were enhanced by the maximum entropy approach (Berger et al., 1996) and implemented the posterior probability definition as a log-linear combination of the set of feature functions (Och and Ney, 2002). Using this technique, it is possible to combine additional feature models in the determination of the translation hypothesis, as shown below (3):

$$
\hat{e}' = \arg\ max_{e'} \left\{ \sum_{m=1}^{M} \lambda_m h_m(f', e') \right\}
$$

(3)

where the feature functions $h_m$ refer to the system models, i.e. BTM, TLM, etc., and $\lambda_m$ represents the corresponding weights of these models.

2.1 N-gram-based translation model

The n-gram-based SMT system operates with bilingual units known as tuples (de Gispert and Martiño, 2002). The tuples are extracted from a word-to-word aligned bilingual corpus according to certain constraints (Crego et al., 2004).

The tuple n-gram translation model determines the joint probability of the source and target language units as shown in Equation (2):

$$
P(e, f) = \prod_{k=1}^{K} P(t_k | t_{k-N+1}, \ldots, t_{k-N})
$$

(4)

where $t_k$ refers to the $k^{th}$ tuple of the given bilingual sentence pair segmented into $K$ tuples and $N$ refers to the n-gram order.

The GIZA++ Toolkit was used to generate word-to-word alignments in source-to-target and target-to-source directions from a bilingual corpus (Och and Ney, 2000). Tuples are then extracted from these alignments. The tuples create a unique segmentation of the sentence pair (Crego et al., 2004). Figure 1 shows an example of tuple extraction from a bilingual sentence pair.

![Figure 1. Tuples from a bilingual sentence pair.](image)

The segmentation is unique and is defined by the word-to-word alignment. The n-gram-based approach is considered to be monotonous.

A detailed description of the SMT system that was used as a baseline in this paper can be found in Mariño et al. (2006) and Costa-Jussà and Fonollosa (2006).
2.2 Additional feature models

A translation system follows the maximum entropy approach and implements the log-linear combination of the BTM and five additional feature models:

**Target language model**
The following equation is used to calculate a standard n-gram TLM:

\[ P_{LM}(t_k) \approx \prod_{n=1}^{K} p(w_n | w_{n-N+1}, \ldots, w_{n-1}) \]  \hspace{1cm} (5)

where \( t_k \) represents the partial translation hypothesis and \( w_n \) is the \( n \)-th word in this partially translated sentence.

The LM representation differs from a state-of-the-art, phrase-based SMT and an n-gram-based translation system in that it is an integrated component of a phrase-based system whereas LM is used as an additional feature in n-gram-based systems as a way of improving translation accuracy.

This additional feature was implemented in the 3-, 4- and 5-gram TLMs in this study.

**Word penalty model**
This feature was implemented in order to compensate for the system’s preference for short output sentences – a phenomenon that is due to the TLM. Technically, the penalization depends on the total number of words in the partial translation hypothesis and can be determined as follows:

\[ P_{wp}(t_k) = \exp(\text{number of words in } t_k) \] \hspace{1cm} (6)

**Source-to-target lexicon model**
This model uses word-to-word IBM Model 1 probabilities (Och et al., 2004) to estimate the lexical weights of each tuple according to the formula below:

\[ P_{ibm}(t_n) = \frac{1}{(I+1)^J} \prod_{j=1}^{I} \sum_{i=0}^{J} p(e_{n,i} | f_{n,j}) \] \hspace{1cm} (7)

where \( f_{n,j} \) and \( e_{n,i} \) are the \( j \)-th and \( i \)-th words in the source and target parts of the tuple \( (e,f)_{n,j} \) and \( J \) and \( I \) are the corresponding total numbers of words on either side of it. Giza++ word-to-word source-to-target alignment was used.

**Target-to-source lexicon model**
This backward lexicon model is the same as the previous model but for the opposite translation direction. We used Giza++ word-to-word target-to-source alignment.

**Word class target language model**
We introduced the novel feature in the final part of the study. A 5-gram model of the word class TLM was used as a method for reducing data sparseness. We considered two types of word classes: a 5-gram model of linguistic classes using part-of-speech (POS) tags and a 5-gram model of statistical classes extracted from the training corpus.

We used the TnT English POS tagger (Brants, 2000) and the FreeLing Spanish tagger (Carreras et al., 2004) for the monolingual corpus tagging (Popović and Ney, 2006).

The target statistical classes were extracted from the training corpus according to the algorithm outlined in (Och, 1999).

2.3 Word reordering

A linguistically motivated word reordering technique was used for Spanish-to-English translation in order to reduce the number of errors caused by the difference in word order between the two languages. A detailed description of the lexicalized reordering procedure can be found in (Costa-jussà and Fonollosa, 2006).

2.4 Decoding and optimization

The MARIE decoder was used as a search engine for the translation system, the details of which can be found in (Crego et al., 2005b). The decoder implements a beam-search algorithm with pruning capabilities. All of the additional feature models described above were taken into account in the decoding process.

Given the development set and references, the log-linear combination of weights can be adjusted by the simplex optimization method (Nelder and Mead, 1965) to maximize the score function (see Eq. 1) according to the highest BLEU score (details can be found in Papineni et al., 2002). Experiments on the log-linear combination of BLEU and NIST scores are planned as part of our future research.
3 TC-STAR evaluation framework

The translation results reported as a baseline system were evaluated in the framework of the TC-STAR 2006 evaluation.

The data provided for shared tasks are from the European Parliament Plenary Sessions (EPPS) Final Text Edition (FTE) data set for English-Spanish and Spanish-English language pairs. The FTE condition corresponds to the official transcripts of the parliamentary sessions and is actually a written language translation condition.

Two reference translations were used for the development and test sets and the first 500 sentences of the official development corpora for both directions were used to maximize the score function. Basic corpus statistics are shown in Table 1.

<table>
<thead>
<tr>
<th>EPPS</th>
<th>Spanish</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>1.3 M</td>
<td>1.3 M</td>
</tr>
<tr>
<td>Words</td>
<td>36.57 M</td>
<td>34.9 M</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>153 K</td>
<td>107 K</td>
</tr>
<tr>
<td>Development set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Words</td>
<td>15 K</td>
<td>12 K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>2.5 K</td>
<td>2.3 K</td>
</tr>
<tr>
<td>Test set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sentences</td>
<td>699</td>
<td>1.155</td>
</tr>
<tr>
<td>Words</td>
<td>31 K</td>
<td>30 K</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>3.9 K</td>
<td>4 K</td>
</tr>
</tbody>
</table>

Table 1. EPPS corpora (M = millions, K = thousands).

4 Experiments and results

The experiments presented in this section are divided into two groups: first we review the impact of TLM threshold pruning on translation accuracy and model size; we then enhance the translation system by incorporating TLM word classes and assess its performance.

4.1 TLM pruning experiments

The LM estimation procedure was performed using the SRI Language Modeling Toolkit (Stolke, 2002) which enables users to set a minimal count of n-grams included in the LM. It is known that the n-gram order, i.e. n-gram history length, has a strong influence on the LM perplexity and the final translation score in a SMT application. We do not thoroughly investigate the impact of the reduction in perplexity, but it is obvious that a significant perplexity reduction will improve the translation accuracy.

We investigated 3-, 4- and 5-gram TLMs in the experiments performed.

The aim of threshold pruning is not to incorporate all of the n-grams that appear in the training corpus fewer times than a cut-off threshold value into the LM. We defined a set of threshold values for each n-gram order (in a “complete” system, the threshold would be 1 for all the n-grams). Obviously the unigram threshold is permanently set to 1 as we do not intend to reduce vocabulary.

Unfortunately, we were not able to perform experiments on the totally unpruned, high-ordered models due to the lack of memory resources and the translation decoder limitations. Consequently, the minimally pruned system configuration includes Threshold 2 for 4- and 5-gram LMs and Threshold 1 for low-order n-grams.

The experiments performed can essentially be considered as an attempt to constrain the n-gram vocabulary in order to reduce system noise and to accelerate the decoding process. The experimental results are shown in Tables 2 and 3.

We analyzed the translations generated by the SMT system and found the results obtained for the development corpus as a result of the model weight optimization (refer to Dev), while the final BLEU scores (case-insensitive) obtained for the test corpora (refer to Test) when the same system configurations and optimized model weights are considered.

SMT decoding can be considered a computationally intensive process in which model size is a crucial factor that can significantly influence the decoding time. The threshold setting considerably reduces the model size, while the case-insensitive BLEU score, which is only associated with the system performance measure in this study, remains constant or even increases.
The next step in our study deals with TLM word class implementation as a method for reducing data sparseness. The feature implements a 5-gram language model of target linguistic and statistical classes.

The tuple translation unit is redefined as a triplet which includes:

- A source sequence of words containing the source side of the bilingual tuple.
- A target string containing the target side of the tuple.
- A class string containing the linguistic or statistical class sequence corresponding to the words in the target string.

This information is only used in the decoding process in order to find an alternative class sequence associated with the competing partial-translation hypothesis. It is not directly used to calculate bilingual translation model probabilities (Mariño et al., 2006).

The TnT English tag set used contains 36 POS tags, while the FreeLing 1.5 Spanish tag set pro-

<table>
<thead>
<tr>
<th>N-gram order</th>
<th>Pruning threshold</th>
<th>BLEU</th>
<th>Model size, millions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2. Final BLEU score for Spanish-to-English translation and pruned target LMs.

<table>
<thead>
<tr>
<th>N-gram order</th>
<th>Pruning threshold</th>
<th>BLEU</th>
<th>Model size, millions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3. Final BLEU score for English-to-Spanish translation and pruned target LMs.
vides greater morphological diversity and 331 different tags. A total of 200 different statistical classes were extracted from the training corpus with the freely available software mkcls (Och, 1999).

Table 4 shows the translation results produced by the SMT systems, including the TLM that incorporates linguistic and statistical classes. The system configuration refers to a baseline.

<table>
<thead>
<tr>
<th>System configuration</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Es-En baseline</td>
<td>65.59</td>
</tr>
<tr>
<td>Es-En POS TLM</td>
<td>65.62</td>
</tr>
<tr>
<td>Es-En Statistical TLM</td>
<td>65.84</td>
</tr>
<tr>
<td>En-Es baseline</td>
<td>56.07</td>
</tr>
<tr>
<td>En-Es POS TLM</td>
<td>56.11</td>
</tr>
<tr>
<td>En-Es Statistical TLM</td>
<td>56.32</td>
</tr>
</tbody>
</table>

Table 4. Effect of the supporting source POS-tags and statistical tags on translation accuracy.

One clear advantage provided by the statistical approach is that the statistical word classes do not depend on the language.

5 Conclusions and future work

In this paper we presented a study of possible simple methods for improving TLM. The baseline system is the translation system used for TC-STAR 2006 evaluation. We investigated two possible methods: simple experiments pertaining to the size of language models; and a linguistically and statistically motivated word class TLM.

There were no significant gains in the reported translation results due to the LM pruning threshold, but there was a significant reduction in the number of stored n-grams.

The main conclusion of this study is that n-gram-based systems are not very sensitive to the decrease in the cut-off value for the appearance of high-order n-grams in the LM. Furthermore, in general, recently appearing n-grams can be discounted to zero as they did not appear in the training corpus.

A more discouraging conclusion is that increasing the history length does not improve system performance. This is explained by the specific character of the n-gram-based system, which includes LM as an additional feature with variable weight.

We will therefore perform further research into this phenomenon in a future study.

The second part of the study introduced a new feature: a TLM based on linguistic and statistical classes. There was no significant gain, but we did observe a slight improvement in performance when Spanish is set as the target language.

There is undoubtedly a great deal of work still to be done in this area. There are already several ideas for statistical and linguistic word class combinations as an additional feature, such as the factored language model (Kirchoff and Yang, 2005).

In contrast to the phrase-based LM factorization, in this study we analyzed the n-gram-based SMT system Moses. Moses is an open-source statistical machine translation system that applies automatically trained translation models to words which may have a factored representation. Factored modeling may be a good solution to the data sparseness problem because it provides a new, intelligent method for combining information sources. We plan to study the use of different morphologically or statistically determined classes as information sources.

Another idea that we intend to cover in future work is to apply the techniques described to the n-gram bilingual translation model, as it is ultimately a language model that deals with bilingual units.

6 Acknowledgments

This work was partially funded by the European Union under the integrated project TC-STAR – Technology and Corpora for Speech to Speech Translation (IST-2002-FP6-506738, http://www.tc-star.org) and by the Spanish Government under grant TEC2006-13964-C03 (AVIVAVOZ project).

References


Abstract
In this paper we show how a single framework for computational modeling of linguistic similarity can be used for solving many problems. Similarity can be measured within or across languages and at various linguistic levels. We model linguistic similarity in three stages: surface similarity, contextual similarity and distributional similarity. We have successfully used the framework for several applications like term normalization, unsupervised shallow morphological analysis, improving information retrieval, transliteration, cognate identification, language identification and sentence alignment etc. For all these applications, we have been able to obtain results comparable with the state of the art.

1 Introduction
Modeling and measurement of linguistic similarity can help in solving many Natural Language Processing (NLP) problems. Linguistic similarity can be within a language or across languages. It is also applicable at various levels: orthographic, phonetic, lexical, syntactic, semantic, etc. There has been a lot of previous work on linguistic similarity (Daga et al., 1994; Lee, 1997; Spasic et al. 2002; Nerbonne and Hinrichs, 2006), especially distributional (Brown et al., 1993; Pereira et al., 1993; Lee, 1999; Lee and Pereira, 1999) and contextual similarity (Dagan et al. 1993; Miller and Charles, 1991; Spasic and Ananiadou, 2005). In this paper we present a single framework for computational modeling of monolingual as well as crosslingual linguistic similarity. Our experiments on using this framework for solving NLP problems have been conducted on English and many South Asian languages. Apart from the framework itself, we have introduced several novel techniques. One of the most important is a computational model of writing systems. So far we have worked more on what we call surface similarity. As far distributional and contextual similarities are concerned, we have, till now, only applied the known techniques for solving some problem in a somewhat novel way.

2 Overview
In our framework, linguistic similarity is calculated in three stages. The first stage is surface similarity, which is calculated mainly by using a Unified Computational Model of Scripts (or Writing Systems) or UCMS. This model consists of component models of scripts, e.g., a Computational Phonetic Model of Scripts (CPMS), a preliminary model of morphology and a computational model of variation. The CPMS itself consists of a model of alphabet, a model of phonology, an aaksharik\textsuperscript{20} model, a Stepped Distance Function (SDF) and an algorithm for aligning strings for calculating linguistic similarity among them.

In the second stage, we model contextual similarity. In one of the techniques for calculating monolingual similarity, each word is modeled in terms of its context. Then these models are compared by

\textsuperscript{20} This term is derived from the word \textit{akshar}, which in common Hindi is a highly ambiguous term. However, in linguistics it has a very long history. In our work we define it as a psychologically real orthographic unit. The closest word in English for akshar is syllable, but there is a difference between the two.
using a similarity measure like the mutual or symmetric cross entropy, which has given very good results for language and encoding identification (Singh, 2006c; Singh and Gorla, 2007) for monolingual as well as multilingual documents. The simplest word model could be a co-occurrence vector. This stage makes use of the information obtained from the first stage.

The third stage is calculation of distributional similarity\textsuperscript{21}. For this, we are using an algorithm based on IBM models (Brown et al., 1993) which uses the estimates of surface similarity obtained from the first stage (Guggilla and Singh 2007).

In this paper we mainly focus on the first stage, i.e. surface similarity. The work on contextual and distributional similarity is ongoing, even though we have tested some techniques for applications like language-encoding identification, translation of tense, aspect and modality markers and sentence alignment of parallel corpora (Singh et al., 2007).

As far as linguistic levels are concerned, our work is restricted to lexical similarity, extra lexical similarity and sentence similarity. Lexical similarity is relevant for a variety of applications ranging from spell checking (which is not very easy for South Asian languages due to lack of standardization etc.) to automatically generating multilingual dictionaries from non-parallel corpora.

Extra lexical similarity (ELS) is applicable to expressions out of which at least one contains one lexical item and at least one contains more than one lexical item. If ELS can be calculated accurately then such expressions, if they occur across languages (which is the more interesting case), can be translated easily, almost as if they were single words. Purely lexical items do not have a one to one mapping across languages. Even if we leave out metaphors and multi-word expressions etc., there are still very commonly used expressions for which there is one ‘word’ in one language but more than one ‘word’ in another language. One of the reasons for this is that some languages are more agglutinative than others. One example of our work on ELS is extraction and translation of multi-word number expressions (Singh, 2007).

Sentence similarity represents how similar two sentences (or segments) are. This kind of similarity can be useful for problems like sentence alignment of parallel corpora (Singh and Husain, 2007).

We are working on using this similarity framework for numerous applications. These include removing bottlenecks for computing in South Asian languages, spell checking, term normalization, identifying rhyming words, a flexible and tolerant input method for South Asian languages, improving information retrieval (Singh et al., 2007), identifying cognate words across languages, identification of language and encoding (Singh, 2006c; Singh and Gorla, 2007) in monolingual and multilingual documents, analysis of spelling/dialectal variation (Singh, 2006d), unsupervised shallow morphological analysis, translation, sentence alignment (Singh and Husain, 2007), comparative study of languages (Singh and Surana 2007), generation of multilingual gloss and quick translation of multi-word entities.

For many of these applications, we have achieved results comparable to the state of the art.

\textbf{Figure 1: Similarity Type Matrix.} Shows pairs of differently similar words. For monolingual similarity, both the words are in Hindi. For crosslingual similarity, the first word is in Hindi and the second in Telugu. Asterisk indicates wrong spelling.

\begin{table}
\centering
\begin{tabular}{|c|c|c|}
\hline
\multicolumn{1}{|c|}{\textbf{Surface}} & \multicolumn{1}{|c|}{\textbf{Contextual}} \\
\hline
\multicolumn{1}{|c|}{\textbf{Monolingual}} & \multicolumn{1}{|c|}{\textbf{Monolingual}} \\
\hline
\textit{Direct} & \textit{Direct} & \textit{Direct} & \textit{Direct} \\
\hline
\textit{Indirect} & \textit{Indirect} & \textit{Indirect} & \textit{Indirect} \\
\hline
\multicolumn{1}{|c|}{\textbf{Crosslingual}} & \multicolumn{1}{|c|}{\textbf{Crosslingual}} \\
\hline
\textit{Direct} & \textit{Direct} & \\
\hline
\textit{Indirect} & \\
\hline
\end{tabular}
\end{table}

\begin{tabular}{|l|l|l|l|}
\hline
\textbf{Items} & \textbf{Surface} & \textbf{Contextual} \\
\hline
\textit{peeli} & [electricity] & \textit{kri} & [book] \\
\textit{baijl} & [electricity] & \textit{paksta} & [book] \\
\hline
\textit{pIA} & [yellow] & \textit{baijl} & [electricity] \\
\textit{pli} & [electricity] & \textit{balha} & [built] \\
\hline
\hline
\textit{naMsa} & [book] & \textit{Ama} & [income] \\
\textit{naMsa} & [book] & \textit{Jai} & [work] \\
\hline
\end{tabular}

\textsuperscript{21} Our use of the terms contextual and distributional similarity requires an explanation. In practice, they are quite often tied together such that we cannot neatly separate the two. When we (in this paper) talk of contextual similarity, the emphasis is on modeling the context, with any method (or no method) being used for modeling distribution. A similar statement can be made for distributional similarity. Most importantly, both can be applied within as well as across languages.
These different kinds of similarity have to be treated differently and they have different applications. We can still build one similarity framework, at least for related languages (Table-1). To calculate some of these similarities, we can use information about other kinds of similarity. For example, for estimating crosslingual similarities, we can use estimates of monolingual similarities.

<table>
<thead>
<tr>
<th>Method</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDS</td>
<td>Model of scripts</td>
</tr>
<tr>
<td></td>
<td>Spell checking, spelling variation, term normalization</td>
</tr>
<tr>
<td>MDC</td>
<td>Model of contextual similarity</td>
</tr>
<tr>
<td></td>
<td>Generating thesauri</td>
</tr>
<tr>
<td>MIS</td>
<td>Model of script along with the model of contextual similarity</td>
</tr>
<tr>
<td></td>
<td>Spell checking, finding out derivationally similar words</td>
</tr>
<tr>
<td>MIC</td>
<td>Model of contextual similarity</td>
</tr>
<tr>
<td></td>
<td>Generating thesauri</td>
</tr>
<tr>
<td>CDS</td>
<td>Model of scripts for different (related) languages</td>
</tr>
<tr>
<td></td>
<td>Finding and studying cognate words</td>
</tr>
<tr>
<td>CDC</td>
<td>Model of crosslingual contextual similarity</td>
</tr>
<tr>
<td></td>
<td>Generating word translations using non-parallel corpora</td>
</tr>
<tr>
<td>CIS</td>
<td>Model of crosslingual contextual similarity</td>
</tr>
<tr>
<td></td>
<td>Finding and studying cognate words</td>
</tr>
<tr>
<td>CIC</td>
<td>Model of crosslingual contextual similarity</td>
</tr>
<tr>
<td></td>
<td>Generating word translations using non-parallel corpora</td>
</tr>
</tbody>
</table>

**Table 1**: Calculating Similarity Types and their Applications.

4 **Surface Similarity**

An important notion in our approach is that of surface similarity. Lexical surface similarity can be divided into two overlapping parts: orthographic and phonetic. It can be monolingual or crosslingual, each of them having different applications. The UCMS (Figure-2) can be used for calculating such similarity. The UCMS originally was an extension of the CPMS (Singh, 2006b), but now the CPMS is one of the component models. At present, the UCMS has been developed only for Brahmi origin scripts used for most of the major South Asian languages like Hindi, Bengali and Telugu etc. which have a coverage of over a billion people. It can be easily adapted for similar scripts like Amharic or Hangul. We are also working on extending the UCMS for other kind of scripts, especially Latin scripts.

![Figure 2: Unified Computational Model of Scripts (UCMS)](image)

5 **Unified Computational Model of Scripts (UCMS)**

5.1 **The Need of Unified Model**

One of the major findings of our work is that knowledge about writing systems can be used for various computational linguistic purposes. Even morphological analysis can become easier and more accurate by using such knowledge. And the best way to use such knowledge is to build a unified computational model of scripts. Such a model not only specifies how information is represented by writing systems, but also includes algorithms for calculating surface similarity based on knowledge about writing systems, morphology and the relation between the two. Such a model will also help us in calculating extra lexical similarity mentioned earlier.

The UCMS can calculate surface similarity, but it can be useful for other purposes too, as it is meant to be a complete model of scripts. To cover various aspects of scripts, we propose ‘smaller’ component models to form a ‘bigger’ model of
scripts. The component models may also be useful individually for certain applications. How many and which component models there are in the unified model as well as the way they are connected to one another will depend on the scripts being modeled. The specific model of scripts that we are proposing is actually a model of scripts classified as abugida scripts, e.g. Devanagari.

The UCMS for Brahmi origin scripts uses the following models, each of which is described in the following sections:

- Model of alphabet
- Aaksharik model
- Phonetic model of scripts
- Preliminary Model of morphology
- Model of variation
- A way of combining these models

<table>
<thead>
<tr>
<th>Feature</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Consonant, Vowel, Vowel modifier, Nukt, Number, Punctuation, Halant, Unused</td>
</tr>
<tr>
<td>Height</td>
<td>Front, Mid, Back</td>
</tr>
<tr>
<td>Length</td>
<td>Long, Short, Medium</td>
</tr>
<tr>
<td>Svar1</td>
<td>Low, Lower Middle, Upper, Middle, Lower High, High</td>
</tr>
<tr>
<td>Svar2</td>
<td>Samvrit, Ardh-Samvrit, Ardh-Vivrit, Vivrit</td>
</tr>
<tr>
<td>Place</td>
<td>Dvayoshthya (Bilabial), Dantoshthya (Labiodental), Dantya (Dental), Varstya (Alveolar), Talavys (Palatal), Mrdhansy (Premaxilla), Komal-Talavys (Velar), Jivhaa-Muliya (Uvular), Svaryantramukhi (Pharynxial)</td>
</tr>
<tr>
<td>Manner</td>
<td>Sparsha (Stop), Nasikya (Nasal), Parshvika (Lateral), Prakampi (Voiced), Sangharshi (Fricative), Ardh-Svar (Semi-vowel)</td>
</tr>
</tbody>
</table>

Table 2: Non-Boolean Phonetic Features

5.2 Computational Phonetic Model of Scripts (CPMS)

Given the similarities among the alphabets of Brahmi origin scripts and the fact that these scripts have phonetic characteristics, it is possible to build phonetic model for these scripts. The correspondence between sounds and letter in Brahmi scripts is almost one-to-one, but not always. We have used a modified version of the phonetic model of scripts proposed by Singh (Singh 2006b). The phonetic model tries to represent the sounds of Indian languages and their relations to the letters. It includes phonetic or articulatory features, some orthographic features, numerical values of these features, and a distance function to calculate how phonetically similar two letters are. The scripts covered by this model are: Devanagari (Hindi, Marathi, Nepali), Bengali (Bengali and Assamese), Gurmukhi (Punjabi), Gujarati, Oriya, Tamil, Telugu, Kannada and Malayalam.

The CPMS itself consists of the model of alphabet, the model of phonology and the SDF. The core of the model of phonology is the definition of phonetic features (Table-2) and the numerical values assigned to them. These features are based on modern phonetics as well as traditional Indian phonetics.

The CPMS assigns a mostly phonetic representation for each ISCII letter code in terms of the phonetic and orthographic features. For example, vowel o and consonant n will be represented as:

176 → [type=v, voiced=t, length=s, svar2=m, svar1=m, height=b]  
198 → [type=c, voiced=t, place=v, manner=n]

Here the feature values are one of those given in Table-2 with the bold letter (e.g., v in Vowel) being used as the code for the value. For boolean features, t and f represent true and false.

![Figure 3: Stepped distance function: various steps differentiate between different kinds of letters. At the end, a quantitative estimate of the orthographic and phonetic distance is obtained.](image)

5.3 Model of Alphabet

The model of alphabet is meant to cover all the alphabets of the related scripts, but it may be more than a superset of these alphabets. By 'model of alphabet' we essentially mean a meta-alphabet, i.e., number of letters and their arrangement, including the basis of this arrangement. It is a conceptual view of the alphabet and also includes a representation based on this view.
Since Brahmi origin scripts have a very well organized alphabet with arrangement of letters based on phonetic features, and also because these alphabets are very similar, it is possible and very useful to have a unified model of alphabet for these scripts. Such a model can simplify computational processing in a multilingual environment.

5.4 Stepped Distance Function (SDF)

To calculate the orthographic and phonetic similarity between two letters, we use a stepped distance function (SDF), which is a part of the CPMS. Since phonetic features differentiate between two sounds (or the letters representing them) in a cascaded or hierarchical way, the SDF calculates similarity at several levels. For example, the first level compares the type (vowel, consonant). There is a branching at the second level and, depending on whether the letters being checked are both vowels or consonants, further comparison is done based on the significant feature at that level: height in the case of vowels and *sthaan* (place) in the case of consonants. At the third level, values of *maatraa* (bound vowel) and *prayatna*, respectively, are compared. Thus, each step is based on the previous step. The weights given to feature values are in the non-decreasing order. The highest level (type) has the highest weight, whereas the lowest level (diphthong, for vowels) has the lowest weight. This process (simplified) is shown in Figure-3.

5.5 Aaksharik Model

Most people categorize Brahmi origin scripts under the heading 'syllabic' or 'alpha-syllabary' or 'abugida'. The reason for this is that these scripts also have syllabic (more accurately, *aaksharika*) characteristics. In other words, the basic orthographic unit in these scripts is an *akshar*.

In our opinion, the subtle difference between syllable and *akshar* can be summarized as:

- **Syllable**: the smallest psychologically real phonological unit
- **Akshar**: the smallest psychologically real orthographic unit

There are two possible definitions of *akshar*. One of them considers *sanyuktashars* (*akshars* that include consonant clusters) to be *akshars*. The other definition does not include *sanyuktashars* among *akshars*, despite the name. We have followed the second definition because it is much more suitable for the current purpose, i.e., calculating surface similarity. However, for display of South Asian language text on screen, the first definition is more suitable. Thus, the *aaksharik* model gives a choice in the way *aksharization* is done. Which one is selected will depend on the application. The model basically specifies a way of grouping letter into *akshars* according to some simple rules (a CFG grammar), but it plays an important part in our unified model. This is not surprising as people who use Brahmi origin scripts, commonly think of *akshar* as a unit. This property of the scripts has been rarely used for language processing.

![Figure 3: Aaksharik Model](image)

5.6 Model of Variation

Model of variation is meant to allow the differences among the scripts-language pairs to be specified systematically. At present this only provides some operations like phonetic or orthographic transformations (Table-3). For example, the transformation operation $y \rightarrow j$ will modify the costs obtained from...
the SDF such that \( y \) becomes close to \( j \). This is because in Bengali, \( y \) is pronounced as \( j \).

### 5.7 Akshar Based FST for Lexicon

For calculating surface similarity, we use akshar as a unit. We extract a word list from the unannotated corpora. This list is then compiled into a dictionary in the form of a finite state transducer (FST) with akshars as the nodes. The structure of the FST currently is restricted to be a trie, but this is more of an implementation issue. Since the number of possible akshar pairs is small, we calculate the costs of all possible pairs using the phonetic model and a modified version of the dynamic time warping (DTW) algorithm (Myers and Rabiner, 1981) used for isolated spoken word recognition (and also for aligning protein sequences, etc.). This algorithm can be used to align two strings made up of nodes. The nodes can be just letters, or they can be feature vectors (as in our case).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monolingual: Hindi</td>
<td></td>
</tr>
<tr>
<td>Integrate</td>
<td>Anusvar ↔ chandrabindu&lt;br&gt;←→ nasal consonant</td>
</tr>
<tr>
<td>Transform</td>
<td>( y \rightarrow e, \ v \rightarrow b, \ s \rightarrow sh )</td>
</tr>
<tr>
<td>Crosslingual: Hindi-Bengali</td>
<td></td>
</tr>
<tr>
<td>Transform</td>
<td>( y \rightarrow j )</td>
</tr>
</tbody>
</table>

*Figure 5: Phonetic and Orthographic Operations*

### 5.8 Preliminary Model of Morphology

Indian languages also have similarities at the morphological level. The (computational) model of morphology of these languages exploits such similarities. We are referring to it as a part of the model of scripts because this model is built on top of the aaksharika model and it may ignore those aspects of morphology (at least for the time being) which don't get directly reflected in written language. In other words, it is a somewhat shallow model to begin with. It is used for doing something more than stemming, but less than complete morphological analysis.

### 6 Calculating Surface Similarity

The method for calculating similarity scores for akshar pairs has been described in the previous section. Once the words have been aksharized and compiled into an FST of akshars, we emulate the DTW algorithm on the FST for calculating surface similarity. This is done by allowing substitution, addition and deletion operations. Substitution is accounted for by simply considering the cost between the akshar on the FST and the akshar in the test word. Deletion means moving forward by one akshar in the test word, but staying on the same node on the FST. Insertion means staying at the same akshar in the test word, but moving to the following nodes in the FST. The search method is based on beam search with cost thresholds.

### 7 Applications

The initial results for some of the proposed applications are quite encouraging. For spell checking and term normalization (mapping spelling variants to one standard form), the precision obtained with a well known and one of the best methods, namely Scaled Edit Distance or SED (Ellison and Kirby, 2006), was 73% and 21%, respectively. These results were for our own implementation of SED. The corresponding figures for the UCMS based method were 66% and 45%. Experiments were also conducted on improving information retrieval (Figure-4) by using estimates of surface similarity based on the UCMS. Precision, recall and F-measure for SED were 66%, 62% and 64%, respectively. The corresponding figures for the method presented in this proposal were 93%, 96% and 95%. The method was also tried for identifying cognates across Indian languages which use Brahmi scripts. The precision obtained by using SED for Hindi-Bengali, Hindi-Marathi and Hindi-Telugu was 24%, 19.5% and 42%. With the proposed method, the precision was 53%, 42.5% and 65.5%. These results exclude proper nouns, which can be matched comparatively easily.

One method for calculating distributional similarity was tried for language and encoding identification. For monolingual identification (Singh, 2006c), the precision obtained varied from 97.64% for 100 bytes of test data to more than 99.50% for documents of sizes of 500 or more bytes. These results are among the best that have been reported for this problem. For language enumeration in multilingual documents (Singh and Gorla, 2007), the precision obtained was 86.93% for correctly identifying two out of two languages, and 96.19% for correctly identifying two out of three languages. Language and identification of individual words was also attempted, which has not been tried so far. For this, the precision for word types was 80.73%
when the languages present in the document were not known, and 90.91% when they were known. The figures for word tokens were 76.82% and 86.80%.

The method for sentence similarity was tried for the problem of sentence alignment. The precision, recall and F-measure figures for 95% confidence were 95.6%, 85.4% and 90.1%. These results were better than those for one of the best previous methods proposed by Moore (Moore, 2007), which were 92.9%, 79.6% and 85.5%.

Evaluation for most of the applications (like sentence alignment and language identification) was performed by preparing randomly extracted and manually labeled reference data. The output was compared against this reference data. In some cases (like cognate identification), the output was manually checked by a person fluent in the concerned language.

8 Future Directions

One interesting theoretical question open for future research is whether it is possible to build one unified model for all the scripts in the world, not just those which are as closely related as the Brahmi origin scripts. If not, then can we at least build a meta-model for all the scripts? This meta-model will, in a way, specify the principles which should be followed for building a UCMS for related scripts. Related to this question is another one: can we find a way to connect together the various models of scripts so that computation across languages with unrelated scripts becomes easier and more precise?

We are working on the principles for creating versions of the UCMS for new scripts and we plan to create such versions for some other scripts. This might involve removing and adding one or two component models. For example, aaksharik model may not be useful for some scripts.

The morphological model used by us is a shallow one in the sense that it only groups words morphologically. We plan to extend this model to include at least morphological segmentation. The results obtained from the morphological model can also be used for increasing the performance of existing rule based (deeper) morphological analyzers.

We are also working on improving models of distributional and contextual similarity and integrating them with the model of surface similarity in a seamless way. We will be able to use the framework for some more applications when we have completed the work on these two stages.

One of the major planned applications is building an integrated digital tool for language resources (Singh, 2006a) so that information from many resources, especially lexical resources, can be made accessible in one place. Calculation of linguistic similarity at various levels will play a major role in this.

9 Conclusions

We have discussed the possibility of using a single framework for calculating linguistic similarity to solve many NLP problems. We have also shown that it is possible to create a unified computational model of many scripts to cover their various aspects. Such a model can be useful for getting better estimates of surface similarity and for solving practical problems. The unified model consists of several interacting component models like a model of alphabet, an aaksharik model, a phonetic model, a model of morphology and a model of variation. The last two of these require more work.

We then used the unified model of scripts along with the algorithm for calculating similarity for some applications, namely spell checking, term normalization, identification of cognate words and improving information retrieval. Our evaluation shows that the results compare favorably with one of the best existing method. We were able to obtain results comparable to the state of the art using the framework described in this paper for several applications.

References


Anil Kumar Singh and Harshit Surana. 2007. Can Corpus Based Measures be Used for Comparative Study of Languages? In Proceedings of the Ninth Meeting of ACL SIG on Computational Phonology and Morphology. Prague, Czech Republic.


RomGen - A Romanian morphological generator, essential component of an English-to-Romanian MT system

Irimia Elena
Research Institute for Artificial Intelligence, Romanian Academy
Bucharest, Romania
elena@racai.ro

Abstract
The aim of this paper is to describe some work (in progress) that we consider a necessary (preliminary) stage to any attempt of constructing a viable machine translation system: the implementation of a morphological generator. The importance of such a tool in a system designed for a language strongly inflected such as Romanian is crucial. We will present here the process of the generator’s implementation, preceded by references to the paradigmatic morphology description of Romanian, which inspired us and offered us important lexical resources.

1 Introduction: some consideration on Machine Translation
There is a large variety of approaches to the machine translation problems, but the most important classification will separate them in two paradigms:
- the rule-based approach, dominating the research field until the end of 80’s, relies on linguistic rules: syntactical rules for analysis and generation, lexical rules for transfer, morphological rules.
- the corpus-based approach, starting from 1989, doesn’t make use of any lexical, syntactic or semantic rule. Its power and effectiveness come from the ability to capture natural linguistic phenomena, using human translation experience from a database of previously translated texts.

Two major directions evolved inside the corpus-based paradigm:
- statistical machine translation, which systems are based mainly on statistical models, with automatically extracted parameters from the distribution of words in large text corpora.
- example-based machine translation, characterized by the use of bilingual corpora as a main knowledge base, at the run-time. The main transfer process implies the matching of the source language fragments from an input text over a database with bilingual examples and the extraction of the target language equivalent fragments, as potential partial translations.

In the context of example-based machine translation paradigm (which is the main theme of my PhD thesis), a very important issue is the acquisition of large repositories of parallel corpora. A significant drawback in having enough and significant data is the rich morphology of a language (as Romanian is), where different example for different morphological forms of the same lemma are necessary. To alleviate the problem of overgrowing databases in the case of an English-to-Romanian example-based translation system, we think that a string matching process at the lemma level should be a better solution than matching the word-forms in the example-database. Obviously, this involves a preliminary lemmatization process of the parallel corpora we intend to use and a subsequently morphologically generation process for the target language. For this purpose, we conceived and implemented a morphological generator for Romanian, which is the subject of this paper and which will be described in the following section.
2 RomGen, Romanian morphological generator in construction

Our approach is based on the paradigmatic morphology theory (Tufis, 1990). The paradigmatic morphology of Romanian has been developed for several years in different formats and variants, the most complete being implemented in the LISP-based ELU linguistic programming environment (Estival et al., 1994). This unification-based implementation of the paradigmatic morphology, together with lexical repositories, associating paradigms and lexical roots to almost 35,000 of Romanian lemmas, was documented in a flat (theory neutral) attribute-value representation (FAVR).

In the context of the paradigmatic morphology theory, a word is treated as an entity made of two fundamental units: a root and an ending (built of one or more desinences and/or suffixes). The root usually carries context-free information, while the ending is a bearer of contextual information. Some contextual information - consisting of restrictions on its use in conjunction with the specified endings - can be associated with the root if there is root alternation (for the same lemma and the same part-of-speech, the different inflected forms can share two or more roots). The information associated with the root is stored in a dictionary (or lexical repository) entry corresponding to the lemma of the corresponding root. Such an entry has the following structure:

```
pos @lemma
root_1 root_2 ... root_k associated_paradigm1
root_k+1 ... associated_paradigm2...
```

The information associated with the ending is stored in ROPMORPH, the file containing a complete inventory of the Romanian paradigms for verbs, nouns, pronouns and adjectives.

Any lemma can be associated to one or more inflectional paradigms. An inflectional paradigm is a tree structure that identifies all the legal endings (and the associated restrictions) which can be associated to a root (or more roots) of a given lemma.

What a morphological generator should be able to do for an MT system is that, having the lemma of a word and the POS-tag corresponding to the needed word form, to produce the correctly inflected form. For Romanian, this means that is necessary to identify the right morphological (inflectional) paradigm in a set of 167 paradigms in the noun’s case and of 72 paradigms in the verb’s case. Accordingly, we face an extremely difficult problem if we try to predict the corresponding paradigm of a word knowing only a POS-tag and a lemma.

Our application has three modules, corresponding to the three following cases for the input lemma:

- The generation module: that can be run only if the lemma is in the lexical repositories, annotated with the morphological root(s) and the morphological paradigm;
- The module for paradigm’s identification: the lemma is not in the paradigmatic dictionary but, together with some (possibly not all) of its inflected forms, it is to be found in the wordform lexicon tbl.wordform.ro
- The module for paradigm’s guessing: the lemma is neither in the paradigmatic dictionary nor in the tbl.wordform.ro lexicon, therefore there is no additional information to help us identify the paradigm(s) to which the word can be associated

2.1 The generation module

In the following example, one can observe a short part from the structure of an entry in the FAVR description of the Romanian paradigmatic morphology:

Example 1. An entry in the FAVR file

```
<PARADIGM PARADIGM="nommascl1"
GEN="masculine" INTENSIFY="none">
<TYPE TYPE="{proper common}"
<NUM NUM="singular">
<ENCL ENCL="no">
<CASE CASE={nominative genitive
dative accusative vocative}>
<TERM TERM="u" ALT = "1"/>
</CASE>
</ENC1>
<ENCL ENCL="yes">
<CASE CASE={nominative accusative}>
<TERM TERM="ul" ALT = "1"/>
</CASE>
<CASE CASE={genitive dative}>
<TERM TERM="ului" ALT = "1"/>
</CASE>
<CASE CASE="vocative"><HUM HUM="imperson">
<TERM TERM="ul" ALT = "1"/></HUM>
<HUM HUM="person">
<TERM TERM="ule" ALT = "1"/></HUM>
</CASE>
</ENCL>
```

```
This example has to be interpreted as following: the first paradigm for the masculine noun receives additional termination (TERM = “u”) for both the proper and common types, for the singular number and the pro-clitical form (ENCL = “no”) in all the cases: nominative, genitive, dative, accusative, vocative. In case of root alternation, the root chosen for the features described is the first in the list of the possible roots (Alt = “1”). The interpretation is similar for the enclitical form, with different terminations corresponding to different cases.

You can observe that an entry has the form of a tree and specifies all the necessary grammatical information for identifying the POS-tag of a word-form (we will use the MSD-tagset, compliant with the Multext-East V3 specifications (Erjavec, 2004) as used by most of our projects and intended to be used for the EBMT system) if we follow a tree branch; the leaves contain information about the termination that should be concatenated to the word’s root to obtain a specific inflected form. The ALT attribute specifies the alternate root (the number represents the position of the root in the list of all possible roots for a lemma and a POS-tag): in Romanian, many nouns have 2 roots (one for singular and one for plural) and the number of roots for verbs can vary from one to seven.

Using these resources, we implemented an application that generates all the word forms – with the MSD-tag associated – for any pair of lemma and POS-tag (n, v, adj) in the lexical files. First, the entry in the lexical repository corresponding to the lemma is identified and the information about the associated root(s) and paradigm(s) are extracted. Then, the corresponding paradigms are identified in the FAVR file and a process of parsing their tree structures is started. Every branch of every paradigm is covered in order to generate the inflected forms and their associated MSD-tags. For the fragment of paradigm presented in the Example 1 and the entry in the lexical repository corresponding to the lemma “maestru” (“master”)

\[ \text{maestru} \text{ Nems-n, maestrul Nemsyv, maestru Nemsyv, maestrul Nemsyv.} \]

2.2 The module for paradigm’s identification

We created a strategy to increase the dimension of the lexical files by identifying the root and the associated paradigm for new words, having as input information as many inflected forms as possible (extracted from tbl.wordform.ro, a collection of word forms that we encountered in manually validated Romanian corpora). If the process of identification succeeds, we can also enrich the tbl.wordform.ro with new word forms.

In the following, we made a description of the algorithm for identification, now being implemented.

**Input data:**

1. \( w_1 \ | \ \text{POS} \)
   \[ w_2 \ | \ \text{POS} \]
   \[ \vdots \ | \ \text{POS} \]
   \[ w_n \ | \ \text{POS} \]

- the list of word forms \( (w_i) \)

2. \( s_1 \ | \ M_1 \ | \ p_1 \)
   \[ s_2 \ | \ M_2 \ | \ p_2 \]
   \[ \vdots \ | \ M_k \ | \ p_k \]

- a list of all the possible suffixes \( (s) \), together with their associated MSD-tags \( (M) \) and paradigms \( (p) \), extracted from the FAVR file.

**Output data:**

1. \( w_1 \ | \ M_1 \ | \ p \)
   \[ w_2 \ | \ M_2 \ | \ p \]
   \[ \vdots \ | \ M_n \ | \ p \]

- the list of the input word forms associated with their MSD tags and with a set of possible paradigms \( P \). \( P \) is a subset of \( \{p_1, \ldots, p_k\} \) and is common to all the word forms in \( \{w_1, \ldots, w_n\} \).

**Description of the identification process:**
1. For each \( w_i \), find the set \( V_i \) of all the triplets \((s_j, p_j, M_j)\), where:
   - \( j \in 1, t \)
   - \( s_j \) is a suffix of \( w_i \), \( s_j \in \{s_1, \ldots, s_k\} \);
   - \( p_j \in \{p_1, \ldots, p_k\} \);
   - \( M_j \in \{M_1, \ldots, M_k\} \).

2. For identifying the root(s) of the set \( \{w_1, \ldots, w_n\} \) we follow the next procedure:
   i. For each \( w_i \), compute the set \( R_{w_i} = \{w_i - s_1 = r_1, \ldots, w_i - s_t = r_t\} \). This means that for each inflected form, we extract a list of roots by eliminating the suffix specific to every applicable paradigm.
   
   \[ R = \bigcap_{i=1}^{n} R_{w_i} \]
   
   ii. Compute the set \( PAR = \bigcap_{i=1}^{n} \text{keys}(PAR_i) \) to find the paradigms applicable to all the inflected forms from the initial group \( L_1 \).

2.3 The module for paradigm’s guessing

For the ultimate situation in which the lemma is not to be found in the extended dictionary, we conceived a procedure to predict a root and a paradigm for the new word, based on the similarities between the unknown lemma ending and the endings for the lemmas in the dictionary (obviously, the searching is made only in the subset of words having the same grammatical category as the lemma’s). We used a dynamic programming algorithm for seeking the longest common substring, adapted for substrings which are endings of the initial strings. For many neologisms and for words with derivational prefixes this module gives good results. Finally, the application requires the human user assistance to assure that the identified paradigms are correct, to choose in case of non-determination and to specify the root (or the roots, in case of alternation). In the following example the results of seeking lemmas similar to the new (inexistent in our resources) Romanian word “fanzin” can be studied:

\[
\begin{align*}
\text{bazin} & n \ zin \ $nomneu1 \\
\text{magazin} & n \ zin \ $nomneu1 \\
\text{mezin} & n \ zin \ $nommasc8 \\
\text{muezin} & n \ zin \ $nommasc8 \\
\text{sarazin} & n \ zin \ $nommasc8
\end{align*}
\]

The user’s linguistic and extra-linguistic (he may know that a “fanzin” is a type of “magazin”) experience will help him/her to choose $nomneu1 as the proper inflectional paradigm for “fanzin”.

Then, the module for generation can be used to produce all the inflected forms of the lemma. Finally, the new word-forms and lemmas are introduced in the lexical resources.

3 Conclusions

This application usefulness was already proved by extending tbl.wordform.ro with 240.464 new word-forms, from which 9149 are corresponding to new lemmas. We also think that this application can be successfully integrated in a question-
answering system for Romanian (another ongoing project, which will develop in a web-service offering to the user the facility to find answers about Romanian legislation).

References


Estival, D., D. Tufis, O. Popescu. 1994. Développement d’outils et des données linguistiques pour le trai-

3 Applications in Natural Language Processing
Discovering grammar rules for Automatic Extraction of Definitions

Claudia Borg
Department of Computer Science and AI
University of Malta
claudia.borg@um.edu.mt

Abstract

Automatic extraction of definitions from text documents can be very useful in various scenarios, especially in eLearning systems. In this paper, we propose an approach aimed at assisting the discovery of grammar rules which can be used to identify definitions, using Genetic Algorithms and Genetic Programming. By categorising definitions to enable the learning of more specialised grammars, we envisage to improve the performance of our learning programs. A genetic algorithm will be used to learn the relative importance of particular predefined features in definitions. To support this algorithm, we also propose a genetic program to evolve new features from existing ones.

1 Introduction

ELearning is a process of acquiring knowledge through electronic aids, by providing students access to materials that will enable them to learn the tasks. The role of the tutor has shifted from the usual role of teaching in a direct manner to one where he manages a collection of learning materials (visual or textual in digital format) and monitors the students’ progress through a Learning Management System.22

Documents normally contain various definitions which describe concepts that are required by students as part of the learning process. Thus, automatic definition extraction can be an important component within the eLearning environment. Definitions contained within a document will help towards the human conceptual understanding of the texts’ meaning, the creation of glossaries and also relate to question answering systems.

The task of definition extraction is a very challenging one. We are trying to identify sentences that contain knowledge about a specific term, which could then be used in applications mentioned above. What’s more, we are attempting to identify sentences which define a term, rather than simply describe it vaguely or compare it to other terms. As in many NLP tasks, the problem could be alleviated by including more information about the text, such as part-of-speech tagging and morphological analysis. Rather than trying to identify arbitrary definitions, one can start by classifying them into different categories, and definition extraction can then be attempted on each category separately to achieve a divide-and-conquer approach to the task. Once all definitions contain the linguistic features and definition categories, definition extraction is more attainable.

In this paper we propose an experiment which combines genetic algorithms and genetic programming to try and discover grammars that could identify definitions present in learning objects. The outcome of this work is to evaluate the use of such machine learning techniques and their results in learning restricted grammars. The grammars developed through these experiments can then be applied to rule-based techniques to extract definitions. The results of the GP and the GA will be used to discover features which identify certain definitions with a high rate of accuracy, but also

22 We will refer to a collection of such materials as a corpus of Learning Objects. However, for the purpose of this work we shall limit the corpus to textual digital documents.
other features to classify the less clear-cut definitions using the features in a combined manner.

The work is carried out in relation to the project LT4eL (Language Technologies for eLearning). The project looks at ways of enhancing the retrieval of learning objects from a LMS by using Language Technologies and Ontologies.

2 Problem Definition

Within a corpus of Learning Objects, texts containing definitions related to the domain of the learning material are normally present. Learning Objects within the LT4eL project have been collected from within the Computer Science and eLearning domains. The objects have been transformed from PDF or Word documents to an XML based format which includes the linguistic annotation with part-of-speech, lemma and morphological analysis information, and retains layout information. Over one thousand keywords and four hundred definitions have been manually identified and annotated for the purpose of keyword and definition extraction tool development and evaluation for each language within the project.

2.1 Defining a Definition

The Oxford English Dictionary defines a definition as a statement of the exact meaning of a word or the nature or scope of something. It normally consists of a Definiendum, the term being defined, and the Definiens, the expression supplying the definition. In order for a definition to capture a word’s meaning, it must describe its operative and functional parts. Context can be used to alleviate ambiguity and reduce the vagueness of a term. Ideal definitions are minimal, integral and talk about the correlation between language and reality. The verb that is used between the definiendum and the definiens is normally referred to as the connector, usually providing the relationship between the two (such as ‘is a’, ‘is called’, ‘means that’).

2.2 Towards a Grammar to Identify Definitions

The linguistic information present in the manually annotated definitions is used as a starting point to identify possible grammar patterns that could constitute a definition. Previous work within this area shows that the use of local grammars which match syntactic structures of definitory contexts are most successful in cases where deep syntactic and semantic analysis is not present (Muresan and Klavans, 2002; Liu et al., 2003). The approach throughout the LT4eL consortium was to develop local grammars for the 9 represented languages (English, Dutch, German, Polish, Bulgarian, Maltese, Czech, Romanian, and Portuguese) to extract definition patterns. An XML transducer, lxtransduce (Tobin, 2005), is used to match the grammar, which conforms to an XML format specified within the tool, by using XPath. When a match is found, a rewrite rule is applied. In our case we wrap the identified definition in an XML tag. The following is an example of a grammar rule which looks for a determiner at the beginning of a sentence followed by a noun:

```xml
<rule name="det_S_noun_phrase">
  <seq>
    <query
      match="s/*[1]@[ctag='DT']"/>
    <ref name="noun_group" mult="+"/>
  </seq>
</rule>
```

The initial induction of grammar patterns is being carried out through human observation. Not only is this task a difficult and tedious one, but it is also prone to human error. We have a number of manually generated grammars which capture definitions, but which are of limited use. Przepiórkowski (2007) reports on the difficulties encountered and the results achieved within the LT4eL project for the Slavic group of languages. By trying to discover grammar rules through manual observation, it becomes problematic to decide which features should be generalised and how to tweak rules not to capture non-definitions. The situation is probably aggravated by the attempt of trying to capture definitions in free structured text rather than texts such as encyclopaedias and dictionaries.

The approach taken by LT4eL was to categorise the definitions and develop grammars separately for each category. The Dutch partner within the LT4eL project analysed if improvement over the grammar is possible through the use of deep parsing (Westerhout and Monachesi, 2007), achieving improvement for definitions containing the verb ‘to
be’ and other connecting verbs such as ‘to mean’, ‘is called’, ‘is used’ among others.

2.3 Road Ahead

It would be desirable to use a machine learning technique that could automatically learn the relevance of possible sequences of the features described above. The experiment will contribute both to the LT4eL project by producing a novel and automatic approach for definition extraction, and to the scientific community providing further application of machine learning techniques within the natural language processing field and offering comparative results to other techniques in definition extraction. For the purpose of this experiment, we will focus on the English corpus only. However, the work proposed should also be extendable to other languages.

3 Proposed Solution

3.1 From Text to Linguistically Annotated XML

Learning objects are usually created by tutors in different formats such as HTML, PDF or other text formats. In order to standardise a common initial format, all objects were converted to HTML. This choice was based on (i) PDF and all proprietary word processors allow documents to be saved as HTML and, (ii) HTML retains layout information such as bold, italic, headings and so on, which is usually important for conceptual understanding.

Whilst retaining the layout information present in the HTML files, linguistic information is added to each token. Using the Stanford part-of-speech tagger (Toutanove and Manning, 2000) and PCKimmo (Antworth, 1990) for morphological analysis, we are able to add the desired linguistic features to the text and reorganise all the information in an XML based format containing this information as meta-data. A tool to process the HTML files to produce the resulting XML files should view the linguistic tools as plug-ins, so as to ensure the possibility of using other linguistic tools available in the future. It would also be possible to add further meta-data should the need for deeper linguistic processing arise. The XML format also conforms to an appropriate DTD, which is derived from the XCES DTD for linguistically annotated corpora (Ide and Suderman, 2002).

Once the corpus has all the linguistic meta-data available, the manually identified definitions can be annotated and thus become machine-readable. Below is an example of a linguistically annotated sentence which has been manually identified as a definition.

\[
\text{<s id="s1501">} \\
\text{<definingText id="dt46" def="m281"}> \\
\text{<markedTerm id="m281" dt="y">} \\
\text{<tok id="t20908" rend="b" ctag="NNP" base="datum" msd="N,SG,proper,vrbl">} \\
\text{Metadata}</tok> \\
\text{</markedTerm> <tok id="t20909" ctag="VBZ" base="be" msd="AUX,PRES,S,finite">is</tok> \\
\text{<tok id="t20910" ctag="VBN" base="define" msd="V,PAST,ED,finite">defined</tok> \\
\text{<tok id="t20911" ctag="IN" base="as" msd="CJ">as</tok> \\
\text{...} \\
\text{</definingText> \\
\text{</s>}
\]

3.2 Categorisation of Definitions

In order to simplify the task, definitions have been categorised into six types. This reduces the complexity of the search space, whereby at each grammar identification attempt we focus only on one type of definition. The types of definitions observed in our texts have been classified as follows:

1. Definitions containing the verb “to be” as a connector.
   E.g.: “A joystick is a small lever (as in a car transmission gearshift) used mostly in computer games.”

2. Definitions containing other verbs as connectors such as “means”, “is defined”, “is called”.
   E.g.: “the ability to copy any text fragment and to move it as a solid object anywhere within a text, or to another text, usually referred to as cut-and-paste.” In this case the term being defined is at the end of the sentence, and it is classified so by the use of “refer to”.

3. Definitions containing punctuation features, usually separating the term being defined and the definition itself.
   E.g.: “hardware (the term applied to computers and all the connecting devices like scanners, modems, telephones, and satellites”
that are tools for information processing and communicating across the globe).” where the definition is contained within brackets.

4. Definitions containing particular layout style. These can include: a term being defined in point form; the use of tables (similar to the punctuation definition, however the term and definition are in separate cells); the defining term as a heading and the definition is the sentence/s below it.

5. Definitions containing a pronoun, usually referring to the defining term which would be placed outside the definitory context. This is common in cases where the definition is over more than one sentence, and the second sentence would refer to the defining term using a pronoun.

E.g.: “This (Technology emulation) involves developing techniques for imitating obsolete systems on future generations of computers.”

6. Other definitions to capture those which do not fall in the above categories.

E.g.: “information skills, i.e. their ability to collect and process the appropriate information properly in order to reach a preset goal.”

The above classification allows us to be able to generalise rules to identify definitions in categories 1 – 5. However, the sixth categorisation does not facilitate the task of identifying a grammar for this category since it contains exceptional cases, and thus cannot be generalised.

3.3 Experiment One: Genetic Algorithm

The availability of manually annotated definitions places us in a favourable situation to attempt learning how to recognize definitions. Given that a corpus contains both a set of definitions and a (usually larger) set of non-definitions, an attempt to learn the importance of features present in definitions is possible. A feature can be seen as a description of characteristics that can help us identify a definition.

A genetic algorithm (Holland, 1975; Goldberg, 1989) is an ideal technique that can be used to learn the importance of the features that can recognise definitions. This can be done by assigning weights to each feature and allowing the algorithm to adjust the weights according to the performance. It also makes it ideal to run the GA on the separate categories of definitions identified in Section 3.2, so that the results can be directed to one given situation at a time.

A feature can be described as a function which given a sentence will return a score. As an example, one feature could describe a part-of-speech sequence that might capture a definition (e.g.: DT→ NN → VBZ → DT → NN → IN → NNS).

The score output from the function will indicate how the sentence rates when the function is applied. The score can be either simply 1 or 0 to indicate a match or non-match, or some numeric value to indicate a level of matching. The latter situation might produce better results by giving more flexibility to the scoring function. For instance, if a sentence would not match the above feature by one pos tag, then the score would be higher than that of a sentence which does not match a single pos tag.

If we have \( n \) features, we would thus have \( n \) functions \( f_1 \) to \( f_n \), each of which takes a sentence and returns a numeric score.

It is important to note that the role of the Genetic Algorithm is not to learn new features but rather to learn the effectiveness of features, as classifiers of definitions.

Before starting the experiment, a predefined set of features will be adopted from the current set of definitions. This collection of features will remain static throughout the experiment. Furthermore, for each such feature we will also have access to a function which will return a score for a given sentence when applied to that feature.

A gene will be a list of length equal to the number of predefined features of numbers. Thus, the \( i \)th gene \((1 \leq i \leq \text{populationsize})\) will have the following structure:

\[
g_i = (\alpha_{i,1}, \alpha_{i,2}... \alpha_{i,n})
\]

Note that \( n \) corresponds to the number of predefined features. The interpretation of the gene is a list of weights which will be applied to the output of the predefined features. Thus, \( \alpha_{i,1} \) is the weight that the gene \( g_i \) would assign to the first feature.
Such a gene will therefore return a score to a given sentence \( s \) as follows:

\[
score_i(s) = \sum_{j=1}^{n} f_j(s) \times \alpha_{i,j}
\]

The initial population will consist of genes that contain random weights assigned to each feature. The interpretation of the gene is a function that when applied to a sentence gives the summation of the feature-function scores multiplied by their weights.

The fitness function will take an individual and produce a score according to its performance. The score will be calculated by applying the gene to both positive and negative examples and will be judged according to how the gene is able to separate the two sets of data from each other.

Crossover and mutation will be carried out in the ‘traditional’ way of Genetic Algorithms. Crossover will take two individuals, bisect them at a random position and create two new children by interchanging the parts of the parents. Mutation will take a random position in the gene and change its value. If the children perform better than the parents, they will substitute them.

Once the population converges, an expected outcome of this experiment is the interpretation of the best gene. The weights produced would give the clearest separating threshold between definitions and non-definitions. It will also allow us to identify which of the features in the predefined feature set are most important, due to a larger weight.

### 3.4 Experiment two: Genetic Programming

Genetic Programming (Koza, 1992) is a technique that uses Genetic Algorithms at its underlying basis, in that it is an evolutionary algorithm. However, Genetic Programming is an optimization problem that evolves simple programs. The main difference between the two techniques is the representation of the population and how the operations of crossover and mutation are carried out. The members of the population are parse trees, usually of computer programs, which are evaluated by the fitness function through their execution. Crossover and mutation are carried out on subtrees, ensuring that the resulting tree would still be of the correct structure.

Whereas the scope of the previous experiment was to learn which are the best performing features from a set of predefined ones for the task of definition extraction, this experiment aims at identifying new features. The initial population will consist of features which were given higher weights by the GA. These features will be translated into parse trees and the score of the fitness function will indicate how well the individual can discriminate between definitions and non-definitions.

The choice of what type of structure we are trying to learn is a determining factor to the complexity of the search space. In our application, two possible options could be regular languages (in the form of regular expressions) or context-free languages (in the form of context-free grammars), the latter having a large search space than the former. Through observations and current work with lxtransduce, regular expressions (extended with a few constructs) would be sufficient to produce expressions that would correctly identify definitions in most cases.

Some of the features used in the first experiment can be used to inject an initial population into the Genetic Program. The selection can be made based on the weights learnt by the GA and translating those features into extended regular expressions. The extensions that are being considered are conjunction (possibly only at the top level), negation and operators such as contains sub-expression. Note that some of these can already be expressed as regular expressions, however, introducing them as a new single operator helps the genetic program learn them faster.

The population will evolve with the support a fitness function in order to select those individuals for mating. The fitness function can apply the extended regular expressions on the given training set and then use measurements such as precision and recall over the captured definitions. Such measurements can indicate the performance of the individuals and will allow us to fine-tune the GP according to the focus of the experiment (where one could emphasise on a high percentage for one measurement at a time, or take an average for both). This flexibility will also allow us to have different results in the various runs of the experiment, where, for instance, in one we could try to learn over-approximations whereas in another we can learn an under-approximation.
Crossover will take two trees and create two new children by exchanging nodes with similar structure. If an offspring is able to parse correctly one definition, it survives into the next generation, otherwise it is discarded. Parents would normally also be retained in the population, since we would not want to lose the good individuals (it is not obvious that their offspring would have the same capability of identifying definitions).

Mutation would take an individual and select at random one node. If that node is a leaf, it is randomly replaced by a terminal symbol. If it is an internal node, it is randomly replaced by one of the allowed operators. Once again, the new tree is allowed to survive to the next generation only if it is able to capture at least one definition.

Once the Genetic Program converges, we expect to have new expressions that would capture some aspects of a definition. The application of this program will allow us to extend our current set of grammar rules by deriving new rules from the above operations. Although we do not expect the genetic program to learn exact rules, it will help towards the discovery of new rules which might have been overlooked, and thus helping towards a more complete grammar for definition extraction.

The GP will also allow the flexibility of running this experiment separately for each of the categories of the definitions as identified in section 3.2. This means that the new features being learnt will be restricted to one category at a time.

3.5 Combining the two experiments

The role of this work is to develop techniques to extract definitions. The two experiments are independent of each other. The GA takes a set of features and assigns a weight to each feature, whereas the GP learns new features through the evolution of the population of extended regular expressions. We can combine the two experiments by migrating the new features learnt by the GP to augment the feature set which is used in the GA.

In the final definition extractor one can start by checking whether a given sentence can be confidently classified as a definition or not by using the features one may learn by running the GP trying to find features which are strict over- or under-approximations. One would then run the weighted sum and threshold as learnt by the GA based on the features we manually identified and others that the GP may have learnt. Clearly the training of the GA would have to be done on a subset of the training set, removing the confidently classified non/definitions. We believe that this approach will improve the quality of the definition identifier.

4 Related Work

Definition extraction is an important task in NLP, and is usually considered as a subtask of information extraction, automatic creation of glossaries, question answering systems.

Work carried out on automatic creation of glossaries usually tends to be rule-based taking into consideration mainly part-of-speech as the main linguistic feature. Park et al., (2001) built a system whereby glossary candidates are identified automatically, ranked and presented to a human expert, who decides whether they should be included within their system. Rules describing the structure of a glossary item are used in their tool and are primarily made up of parts-of-speech. However, further linguistic tools are placed in a pipeline architecture to fine tune the results based on new linguistic knowledge at every step. Their work was applied to technical texts in the automotive engineering and computer help desk domains. In this type of corpora, glosses are usually well-structured and thus easier to identify using static rules than in a more generic domain. We envisage that the pos structures identified in this work may provide us with features to use in our experiments.

DEFINDER (Klavans and Muresan, 2000) works towards the automatic extraction of definitions in well-structured medical corpora, where 60% of the definitions are introduced by a set of limited text markers (such as ‘-‘, ‘()’). Further work (Klavans et al., 2003) looks at the Internet as a corpus, focusing mainly on large government websites, trying to identify definitions by genus. In this task, several problems are identified, both in format and in content. Definitions can be ambiguous, uncertain or incomplete, which have also been encountered in our project. Another problem encountered is that the Internet is a dynamic corpus, and different websites could change their information over time. It also describes decision rules for particular cases, which could be applied to the features in our experiments. An interesting discussion is presented in how to evaluate a definition extractor, proposing a Gold Standard for such a type of
evaluation, based on qualitative evaluation apart from the standard quantitative metrics.

Fahmi and Bouma (2006) tackle the problem of definition extraction using an incremental approach, starting with individual words, then adding syntactic features etc. They look at the potential definition sentences that fall into our first category (containing the verb to be) from a Dutch corpus of medical articles extracted from Wikipedia. These sentences are manually annotated as definitions, non-definitions and undecided, and this corpus of sentences is used as their training and evaluation data for the experiments carried out. They identify several attributes that could be of importance to the experiments, namely text properties, sentence position, syntactic properties and named entity classes. Learning-based methods are then used to identify which of these features, or combination of, would provide the best results. These feature combinations can also be considered for the experiments described above. A difference between our work and theirs is that we primarily will be using our learning algorithm to learn more generic definitions. However, we plan to run our algorithm also on well-structured, technical texts in order to see the effect of structure in the corpora on the quality of results and also in order to be in a position to compare results to those presented by Fahmi and Bouma.

Identifying grammars for the task of definition extraction can be related to the area of Grammatical Inference. In this field, research attempts to use machine learning techniques to learn grammars and language form data. A variety of applications can be identified including syntactic pattern recognition, computational biology, natural language acquisition, data mining and knowledge discovery. Genetic algorithms and genetic programming are two related machine learning techniques which have been applied to grammatical inference.

Work carried out by Smith and Witten (1995) describes a GA that adapts a population of hypothesis grammars towards a more effective model of language structure, with emphasis on inferring practical natural language grammars. The lexical categories are also discovered through the learning process. Of particular interest are the discussions of how constructs such as AND, OR and NOT are used in the representation of the individuals, and how the fitness function, crossover and mutation are selected and applied. These type of constructs are similar in function as to what we propose in Section 3.4 as part of the extended regular expressions.

Similarly, Lankhorst (1994) describes the inference of Context-free grammars through the use of a Genetic Algorithm. The GA is tested to learn languages varying complexity, starting from a rather trivial grammar (equal brackets) to a micro-NL grammar. The former grammar was successfully learnt by the GA, however, the latter was learnt only after the fitness function was changed. This reinforces the importance of such decisions, and the fitness functions described will be taken into account. Loose (1994) attempts to learn syntactic rules and tags with a GA. Again, many components of this work are of interest to us and will be taken into consideration (fitness function, crossover, mutation). However, the most interesting comment is a concluding remark, in that “if one is willing to assume that parts-of-speech are known accurately, the learning of syntactic rules can occur at a much higher rate...”. This is encouraging towards the possible achievement of positive results in the experiments described above.

5 Conclusions

We have described the motivations behind definition extraction, difficulties encountered and proposed a possible solution in the form of an experiment. To our knowledge, the use of Genetic Algorithms and Genetic Programming has not been used in the way being proposed for this experiment. The results should be of interest not only to the natural language task of extracting definitions, but also to the machine learning task of combining GAs with GPs.

References


http://www.cs.vassar.edu/XCES/


SketchNet: Knowledge Based Word Sense Disambiguation

Alan Duca
University of Malta
Department of Computer Science and Artificial Intelligence
Msida, Malta
aduca@keyworld.net

Abstract
This synopsis presents a new knowledge based approach to the inherently difficult problem of automatic sense disambiguation. The proposed method makes use of the semantic richness of polysemous words to overcome ambiguity. The approach is implemented in SketchNet, a .Net application capable of performing target word sense disambiguation. When evaluated on a test set of nouns with very fine sense distinctions, the proposed method ranked the correct sense in the top two places on 80% of occasions.

1 Introduction
Automatic Word Sense Disambiguation (WSD) can be defined as the process of automatically determining the correct sense of a polysemous term in a given context. A term is said to be polysemous when it can be used to refer to a number of different concepts. For the purposes of this project, word senses and concepts have a one-to-one correspondence.

It can be argued that one of the main contributors to ambiguity is polysemy. This seems to make sense, since ambiguity often arises due to a single value pointing to a number of different meanings. However, the highly abstract nature of the notion of sense itself makes this a very difficult area to draw conclusions about. The same notion of sense has been a debated issue since the beginning of philosophical thought, and still persists as a concept for which no clear and universally accepted definition exists (Ide and Veronis, 1998).

Nevertheless, WSD has been a major problem in several language processing tasks since it was first identified. The large frequency of polysemous words in the human language has caused (and continues to cause) great difficulties in language processing tasks such as text-to-speech technologies, information extraction and, most notably, machine translation (MT). Indeed, automatic WSD was first acknowledged as a problem in its own right in 1960 by Bar-Hillel, who was actually working on MT; at that time, a science in its infancy (Ide and Veronis, 1998; Mihalcea and Pedersen, 2005; Bar-Hillel, 1960).

The lack of clarity with respect to the notion of sense was also not enough to thwart the wave of scientists and researchers who followed up on the remarks of Bar-Hillel. Since the 1960s, numerous methods have been proposed, attempting to tackle the daunting task of automatic disambiguation.

To this day, WSD remains one of the most open areas in natural language processing, partly because of the fact that little progress has been made over the span of a half-a-century. The performance of even the best automatic disambiguation methods leaves much to be desired, and the methods which do provide good results are often not very scalable.

2 Approaches to WSD
At the time of writing, the most recent, all-inclusive, review of WSD methods is the one carried out by Rada Mihalcea and Ted Pedersen entitled “Advances in Word Sense Disambiguation”. This very interesting tutorial was given in 2005 during an AAAI conference, and investigates the
landmark WSD methods developed since the identification of WSD as a problem. Mihalcea and Pedersen identify four types of disambiguation techniques:

1. Rule Based methods
2. Supervised methods
3. Unsupervised methods
4. Knowledge Based methods

Rule Based methods were the first methods to be developed, and involved the manual creation of knowledge sources. Such methods would rely on the rules represented in these knowledge sources to perform disambiguation. Problems such as the “knowledge acquisition bottleneck”, however, made these approaches expensive to pursue (Ide and Veronis, 1998).

Supervised methods of disambiguation make extensive use of semantically annotated corpora to teach an algorithm how to disambiguate correctly. These methods were particularly prevalent in the 1990’s. Many of these approaches involved a “bag of words” technique, by which the words in the immediate context of a semantically annotated polysemous term are recorded during training, and then matched to the context of a word to be disambiguated during testing. Though these methods often produced good results, they were almost never scalable and always very domain specific. This resulted from the difficulty in manually annotating the senses of words in a corpus.

Unsupervised methods of disambiguation perform “word sense discrimination” rather than “word sense disambiguation”. Methods adopting this approach do not assign a sense to a polysemous term occurrence. Instead they attempt to cluster the occurrences of a polysemous word based on their apparent sense. It is then left to the user to “label” the resulting clusters.

Knowledge Based approaches represented a turning point in WSD history (Ide and Veronis, 1998). These methods came about in the 1980’s, after large-scale knowledge sources such as dictionaries, thesauri, corpora and semantic networks became widely available. Knowledge Based methods would make use of these readily available sources of knowledge to perform automatic disambiguation. Different knowledge based approaches differ in the number of external knowledge sources they use, which ones they use, and how they make use of them. Such methods have the advantage of requiring no human intervention, and being easily scalable.

The method proposed in this synopsis and implemented as SketchNet adopts a knowledge based approach. It makes use of two external knowledge sources: WordNet and the Sketch Engine; both of which will be briefly described in the following sections.

3 WordNet

One of the major contributors to knowledge based approaches of automatic sense disambiguation is undoubtedly WordNet.

WordNet is a semantic lexicon developed by George Miller and his colleagues at Princeton University. It can be seen as a cross between a dictionary, thesaurus and semantic network.

Concepts in WordNet are organized into synsets. A synset is a set of terms referring to the same concept. Rather than defining terms, like most dictionaries would, WordNet defines synsets. Synsets are organized into taxonomies, linked together through a number of semantic relationships. These relationships include:

- **Hypernymy**: IS-A relationship in the “child to parent” direction.
- **Hyponymy**: IS-A relationship in the “child to parent” direction.
- **Holonymy**: the PART-OF relationship in the “part to whole” direction.
- **Meronymy**: the PART-OF relationship in the “whole to part” direction.

WordNet includes synsets formed from all types of words, but significantly structured semantic networks are available only for noun and verb synsets. Synsets composed of other classes of terms either belong to a far less complex taxonomy, or are not linked to other synsets at all.

The WordNet database of synsets and semantic relationships is completely hand crafted, and undergoes constant enhancement. The granularity of senses expressed in WordNet has been criticized as being too fine for most linguistic purposes, but this has not discouraged several researchers from making use of the WordNet database in their efforts of producing solutions to the WSD problem (Ide and Veronis, 1998).
4 The Sketch Engine

Strictly speaking, the Sketch Engine (SkE) is not a knowledge source in its own right. Instead, it is a system which makes use of a knowledge source: a large POS-tagged corpus. In fact, SkE is an enhanced Corpus Query System (QCS) (Kilgarriff et al., 2004).

The SkE was created by Adam Kilgarriff, Pavel Rychly, Pavel Smrz and David Tugwell. Its aim is to provide easier access to the behavioral information of a particular term in large corpora. While most QCS offer only basic access and querying functionality to information within corpora, SkE offers detailed summaries of the grammatical behavior of a term in a corpus. It does this in the form of word sketches.

A word sketch is defined as a one-page automatic, corpus-based summary of a word’s grammatical and collocational behavior. Word sketches are generated by SkE upon the request of the user, provided that it is supplied with a query consisting of a term in lemma form and its corresponding part-of-speech. Such word sketches illustrate to the user which words are found to be grammatically related to the queried term, and through which grammatical relationships. The grammatical relationships recorded by SkE in word sketches include:

- Object-of
- Has-object
- Modifiers
- Modifies
- Subject-of
- Has-subject

Word sketches are divided into tables, each representing some particular grammatical relationship the sketched (or queried) word has been found to participate within. These tables would contain the terms discovered to share the represented grammatical relationship with the sketched word. Appended to the terms found to co-occur with the sketched word, one would also see the number of times the same co-occurrence was observed in the analyzed corpus.

SkE relies on a small set of rules, in the form of regular expressions over part-of-speech tags, to identify the grammatical relations recorded in a word sketch. This makes the generation of word sketches not only fast and efficient, but also very scalable.

5 The Proposed Method

When studying the behavior of words in the 1940’s, Zipf and Thorndike made an interesting observation: the number of synonyms a word has is an indication of its semantic richness (Zipf, 1945; Thorndike, 1948; Ide and Veronis, 1998). In other words, the more synonyms a word has, the more polysemous it is likely to be.

Synonymy is an interesting concept, since it appears to be the opposite of ambiguity. While ambiguity entails a single term referring to one of a number of possible concepts, synonymy involves one concept being describable by more than a single term. Figures 1 and 2 below illustrate the notions of term ambiguity and synonymy as assumed in this project.

If we were to “super-impose” both of the figures, we would be presented with an interesting property of semantic richness. Figure 3 outlines this feature.

Suppose that, in the context of Figure 3, the polysemous word “m” occurs in a sentence, and we would like to disambiguate it. If we manage to

![Figure 1](image1.png)

**Figure 1:** This illustration shows word ‘m’ referring to 3 different concepts ‘a’, ‘b’, and ‘c’. This one-to-many behavior is what makes words ambiguous.

![Figure 2](image2.png)

**Figure 2:** This illustration shows 3 different words ‘m’, ‘n’ and ‘o’ referring to the same concept ‘c’. This many-to-one behaviour is better known as synonymy.
show that words “n” and “o” are likely to appear in the same sentence instead of “m”, then it would be an indication that word “m” is probably referring to concept “C”. Therefore, by making use of the synonymy of a polysemous word, it would be possible to identify the concept being referred to by the polysemous word. Given the semantic richness of polysemous words, this has the potential of being a good disambiguator.

The issue now is determining a way of showing whether words “n” and “o” are likely to occur instead of “m” in some sentence. This can be done through the use of word sketches, described in the section about Sketch Engine. The procedure follows the steps explained below:

1. Identify the synonyms of one sense of the polysemous word to be disambiguated (Y).
2. Identify any grammatical relations in which the word to be disambiguated participates (G).
3. For each grammatical relation (g; g C G) identified in the previous step, do steps 4-7.
4. Identify the term related to the ambiguous term via relation g (x).
5. Identify the inverse relation corresponding to relation g (g⁻¹).
6. Obtain the word sketch of related term x from SkE (S(x)).
7. From S(x), obtain the list of terms found to co-occur with x via the inverse relation g⁻¹. Add the obtained terms to a list of terms (L).
8. Intersect the list of terms L with the list of synonyms Y built in step 1, and count the number of common elements. This provides the “incidence count” for the chosen sense of the word to be disambiguated.

The entire process described above is repeated for each sense of the polysemous word to be disambiguated. This results in an “incidence count” value for each sense of the word to be disambiguated. The sense of the polysemous word which yields the greatest incidence count value (i.e.: has most elements in common) is chosen as the correct sense.

It is important to note that, however, when used alone, synonyms are often not enough to act as features defining a concept. Given that the list of synonyms for some word can sometimes be very limited, it would be required to obtain a larger and more clear picture of the various senses of a polysemous word, especially if we intend to disambiguate it. Lack of clear enough pictures, caused by the small number of synonyms, may cause the method to frequently fail. This can be overcome, however, through the use of semantic signatures. A semantic signature is a list of words which are semantically related to the subject term. These signatures are obtained from WordNet, by collecting the words present in the synsets connected to the concept to be disambiguated through semantic relations such as hypernymy and hyponymy. Semantic signatures would replace the synonym lists built in the first step of the detailed procedure, adding more body to the features identifying each sense of the polysemous word to disambiguate. This would eventually yield more consistent results.

6 Sample Task

In this section, we attempt to disambiguate the polysemous noun “gas” in the sentence “Step on the gas!” using the method described in the previous section.

We start by identifying the possible senses of the term “gas”. WordNet yields six different senses for the noun “gas”:

<table>
<thead>
<tr>
<th>Sn.</th>
<th>WordNet Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The state of matter distinguished from the solid and liquid states by: relatively low density and viscosity.</td>
</tr>
<tr>
<td>2</td>
<td>A fluid in the gaseous state having neither independent shape nor volume and</td>
</tr>
</tbody>
</table>
being able to expand indefinitely.

3 A volatile flammable mixture of hydrocarbons.

4 A state of excessive gas in the alimentary canal.

5 A pedal that controls the throttle valve.

6 A fossil fuel in the gaseous state.

Next, we make use of WordNet’s semantic relations to obtain a semantic signature for each sense of the word “gas”. The following table contains parts of the semantic signatures obtained by traversing WordNet’s synset taxonomy for each sense of “gas”:

<table>
<thead>
<tr>
<th>Sn.</th>
<th>Semantic signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>gaseous, state, matter, form, liquid, solid</td>
</tr>
<tr>
<td>2</td>
<td>fluid, ichor, liquid, atmosphere, oxygen</td>
</tr>
<tr>
<td>3</td>
<td>gasolene, gasoline, petrol, fuel, biomass, propane, kerosene, napalm</td>
</tr>
<tr>
<td>4</td>
<td>flatulence, flatulency, state, condition, acathexia, asphyxia</td>
</tr>
<tr>
<td>5</td>
<td>accelerator, throttle, pedal, gun, treadle, foot, lever, clutch, brake</td>
</tr>
<tr>
<td>6</td>
<td>natural, fossil, coal, petroleum, crude</td>
</tr>
</tbody>
</table>

We proceed to identify grammatical relations (in the target/input sentence) within which the term “gas” participates. We also identify the terms related to the word “gas” through these relations. In this case, given the input sentence “Step on the gas!”, it is evident that the verb “step” is related to the term “gas” through the preposition “on”. This kind of relationship is listed in SkE word sketches as a “pp_obj_on-p” relation.

The next step involves identifying the grammatical relations corresponding to the inverse of the grammatical relations found. Since only one grammatical relation has been identified in this scenario, we will have only one corresponding inverse relation. The inverse of the identified relation “pp_obj_on-p” is listed in SkE word sketches as “pp_on-p”.

As detailed in the previous section, we then query SkE for the word sketches of terms found to be related to the term to be disambiguated. In this case, we query SkE for the word sketch of the verb “step”.

We then proceed to obtain the part of the retrieved word sketch which represents the respective inverse relation. In this case, we are interested in the table representing the “pp_on-p” relations of “step”. Part of this table is shown in Figure 4 below:

<table>
<thead>
<tr>
<th>pp_on-p</th>
<th>Incidence count</th>
</tr>
</thead>
<tbody>
<tr>
<td>newsman</td>
<td>19.66</td>
</tr>
<tr>
<td>accelerator</td>
<td>19.22</td>
</tr>
<tr>
<td>gas</td>
<td>16.21</td>
</tr>
<tr>
<td>toe</td>
<td>16.03</td>
</tr>
<tr>
<td>scale</td>
<td>14.06</td>
</tr>
<tr>
<td>stage</td>
<td>12.31</td>
</tr>
<tr>
<td>mine</td>
<td>12.04</td>
</tr>
<tr>
<td>drawing-pin</td>
<td>10.49</td>
</tr>
<tr>
<td>lilliputian</td>
<td>10.44</td>
</tr>
<tr>
<td>foot</td>
<td>10.32</td>
</tr>
<tr>
<td>train</td>
<td>10.2</td>
</tr>
<tr>
<td>landmine</td>
<td>9.63</td>
</tr>
<tr>
<td>jugular</td>
<td>9.51</td>
</tr>
<tr>
<td>stick</td>
<td>9.23</td>
</tr>
<tr>
<td>tiptoe</td>
<td>8.93</td>
</tr>
<tr>
<td>rug</td>
<td>8.03</td>
</tr>
<tr>
<td>cobra</td>
<td>7.65</td>
</tr>
<tr>
<td>rake</td>
<td>7.62</td>
</tr>
<tr>
<td>sleeper</td>
<td>7.42</td>
</tr>
<tr>
<td>floorboard</td>
<td>7.34</td>
</tr>
<tr>
<td>frock</td>
<td>7.04</td>
</tr>
<tr>
<td>magnet</td>
<td>6.92</td>
</tr>
</tbody>
</table>

Figure 4: Part of the table representing the “pp_on-p” relation for the sketched term “step”. The listed terms are those identified by SkE as sharing the represented relation with the sketched term. The underlined integer represents the number of times the term-relationship occurrence was observed by SkE in the examined corpus.

Finally, we intersect the terms obtained from the “pp_on-p” table of the word sketch for “step” (partly shown in Figure 4) with the terms within the semantic signatures for each sense of the noun “gas”. Hence, we obtain an incidence count value for each sense of the ambiguous term. The following table illustrates the yielded results:

<table>
<thead>
<tr>
<th>Sense</th>
<th>Incidence count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Lever(1) + Accelerator(4) +</td>
</tr>
</tbody>
</table>

73
Based on these results, we have shown that the terms within the semantic signature of the 5th sense of “gas” (i.e.: A pedal that controls the throttle valve) are the most likely, out of all the semantic signatures, to occur in the same sentence instead of “gas”. Since the proposed method concludes by choosing the sense which yields the highest incidence count as the correct sense, in this case, we choose the 5th sense as the correct one for “gas” in the given context. This also appears to be, in reality, the most obvious or probable sense that the term “gas” would be alluding to given the same context sentence.

Note that only sense 5 produced an incidence count greater than zero. This happened because only the semantic signature of the 5th sense of “gas” had any common elements with the list of terms obtained from the word sketches of terms found to be related to it in the input sentence. In fact the chosen sense of “gas” was found to have 4 common elements with this list of terms.

Note also that some common elements provided an incidence value of more than 1. This is because they were observed more than once by SkE (see section 4 on the SkE).

7 SketchNet

SketchNet is a .Net application which implements a version of the proposed method. It was developed to serve mainly as a means of evaluating the method discussed in this synopsis. However, SketchNet also allows the user to monitor the progress of the method while it is being executed, while providing him with useful information related to the operation being carried out. Last, but not least, SketchNet allows the modification of certain parameters which would affect the outcome of the disambiguation process.

Since SketchNet makes use of the WordNet database, it requires an installation of this on the host machine. It also requires an internet connection and subscription to the services of SkE, allowing it to make use of the latter.

Given an input text, SketchNet attempts to apply the proposed method to disambiguate a single term, marked as “ambig” (i.e.: ambiguous). The input text must be POS-tagged in the C5 tagset, allowing SketchNet to identify any grammatical relations present in the input text in a way not dissimilar from that used by SkE itself.

Once the disambiguation process has been completed, SketchNet proceeds to display the sense chosen as correct, and rank the remaining senses of the ambiguous term.

Using SketchNet, one can specify certain parameters, such as which WordNet relations to traverse (and how deep traverse them) when building the semantic signatures for each sense. Modifying this parameter, and others like it, are sure to affect the outcome of the disambiguation process. The underlying method remains, however, completely unchanged.

8 Results

The proposed method was tested, through the use of SketchNet, on a compiled test set. The test set used included 5 ambiguous words: 3 nouns and 2 verbs. For each ambiguous word tested upon, 10 different cases were made available; each case consisting of a sentence within which the ambiguous word occurred. These sentences were randomly chosen from semantically annotated texts within SemCor 2.1.

When tested on nouns, SketchNet produced the following results:

<table>
<thead>
<tr>
<th>Word</th>
<th>No. of senses</th>
<th>% top ranking correct</th>
<th>% top 2 ranking correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass</td>
<td>8</td>
<td>50%</td>
<td>80%</td>
</tr>
<tr>
<td>Earth</td>
<td>7</td>
<td>30%</td>
<td>80%</td>
</tr>
<tr>
<td>Plant</td>
<td>4</td>
<td>30%</td>
<td>80%</td>
</tr>
<tr>
<td>Avg.</td>
<td>6.33</td>
<td>36.66%</td>
<td>80%</td>
</tr>
</tbody>
</table>

During testing, SketchNet used a “fall-back” mechanism, reverting to the most common sense of the term being disambiguated in case it failed to complete the disambiguation process at hand. This,
however, was only required in around 5% of the test cases carried out.

9 Difficulties of WSD Evaluation

If one were to count the number of different disambiguation methods developed since the 1960s, and then count the number of different evaluation methods which have been used since then, one would most probably find out that there exists an almost equal amount of both. Evaluation of a WSD method is no trivial task (Ide and Veronis, 1998; Mihalcea and Pedersen, 2005; Resnik and Yarowsky, 1997). Difficulties include:

- Lack of availability of semantically annotated sample corpora.
- Difficulty of manually annotating text semantically, especially when very fine sense distinctions are involved.
- Need to have semantically annotated texts in which senses are tagged relative to the utilized knowledge source or the referenced set of senses.

All these difficulties, along with many others, have forced researchers working in this area to build their own testing sets. Consequently, very often these testing sets are small and do not easily win much credibility. To this day, the world of WSD still lacks a “standard” test mechanism, or test set, useable by any disambiguation method.

It is also difficult to compare one disambiguation method to another. Though one method might provide better results than another, it may be the case that the better method does not disambiguate using sense distinctions as fine as the other method does. It may also be the case that the better method makes use of annotated corpora, while the other method foregoes such procedures; or that the better method is far less scalable and generally applicable than the other. Given the diversity of disambiguation methods developed up to now, one must take many factors into consideration when making comparisons, as it is clear that the benefits in scalability, cross-domain applicability and other factors may outweigh performance increase.

10 Method Evaluation

When applied through the use of SketchNet on a small testing set, the proposed method provided the results listed in section 8.

Though, in the case of nouns, an average of 36.6% correct results obtained when considering only the top ranked sense does not seem impressive, it must be noted that disambiguation is being performed given very fine sense distinctions. If we were to rely on chance to do the same task, we would have ended up with around 17% accuracy considering only the top ranked sense. It must also be noted that in the case of disambiguation of nouns, an average of 80% correct results were obtained when considering the top 2 ranked senses. This would compare to an average of 34% accuracy when relying on chance to obtain correct disambiguation.

It is clear that when disambiguating nouns, the method performs much better than when disambiguating verbs. The reason behind this becomes evident if one were to compare WordNet’s semantic network for nouns to that for verbs: the semantic network for nouns is far more rich and densely connected than that for verbs. This makes semantic signatures built for noun terms much stronger than those built for verb terms. Being a knowledge based approach, the proposed method has the “deficiency” of relying to a great extent on the knowledge sources it makes use of, inheriting much of the issues the latter suffer from.

An encouraging fact is that the existing disambiguation mechanism presented here through SketchNet can be greatly improved upon. SkE, for example, was not specifically designed for WSD purposes. Therefore, if one were to develop a word sketching mechanism which would focus on recording word behaviorisms more relevant to the task of disambiguation, the proposed method would provide better results. Enlarging the raw corpus used by SkE to produce word sketches may also positively affect performance.

One must also bear in mind that absolutely no semantically annotated corpora were required to produce the results discussed here. This fact, coupled with the facts that the method was applied to texts from cross-domain documents and that disambiguation was performed given very fine sense distinctions, makes the proposed method an interesting approach to consider further work upon.
11 Related Work

Knowledge-based automatic disambiguation methods that share some common elements with the method proposed in this synopsis do exist. Many knowledge-based methods do, in fact, make use of WordNet’s semantic networks for nouns and verbs, though this method is the first one to make use of word sketches provided by SkE.

One such method is Resnik’s Information Content method, presented in 1995 (Resnik, 1995). Resnik followed in the work of Sussna, who, two years earlier, had produced an automatic disambiguation method which makes use of WordNet’s noun semantic network to disambiguate targeted polysemous nouns (Sussna, 1993).

In Sussna’s method, the “link-distance” between each sense of the noun to be disambiguated and each sense of some other noun occurring in the immediate context is counted. This “link-distance” corresponds to the number of relations one has to traverse in the WordNet semantic network to get from one synset (representing some sense of the word to be disambiguated) to the other (representing some sense of the word co-occurring in the immediate context). The two senses yielding the shortest “link distance” value are chosen as the correct senses.

Resnik improves on this method by introducing the notion of “information content” to the equation. “Information Content” states that the more abstract, or general, a concept is, the less information value it has. Resnik therefore argues that semantic links deep within the WordNet hierarchy are essentially “shorter” than those found higher up in the hierarchy, since the former link concepts which possess a greater amount of information to share in common.

Other interesting methods which share common elements with the one proposed here exist, such as the one by Rada and Moldovan, based on “Semantic Density” (Mihalcea and Moldovan, 1998), but will not be described here. Suffice to say, most make use of WordNet in some way or another as a readily available knowledge source.

References


George Zipf, 1945. “The meaning-frequency relationship of words”.


Philip Resnik, 1995. “Using information content to evaluate semantic similarity in a taxonomy.”

Philip Resnik and David Yarowsky, 1997. “A Perspective on Word Sense Disambiguation Methods and Their Evaluation”.


F-structure: Derivation-based construction and constraints

Miltiadis Kokkonidis
Computational Linguistics Group
University of Oxford
miltiadis.kokkonidis@clg.ox.ac.uk

Abstract

Type-Logical Lexical Functional Grammar is a new syntactic framework combining the derivational nature of Type-Logical Categorial Grammar with a distinct Lexical Functional Grammar flavour while dispensing with syntactic categories, a key ingredient in both TLCG and LFG. The derivational nature of TL-LFG comes from the use of its type system for syntactic analysis and semantic composition. The LFG flavour comes from both the emphasis on functional structure and the functional constraints. TL-LFG is powered by little more than a type system based on the implicative fragment of intuitionistic linear logic with first-order unification; its other major piece of formal equipment is a simple regular-expressions checking mechanism. This paper explains the synergy between the two.

1 Introduction

The prime motivation for a theory such as LFG (Kaplan and Bresnan, 1982) is evidence that functional structure is a far better candidate for expressing crosslinguistic syntactic generalisations. Furthermore, syntactic functions prove to be valuable primitives for successful description on complex syntactic phenomena in a fairly intuitive manner.

Glue Semantics (Dalrymple et al., 1993; Dalrymple, 1999; Dalrymple, 2001) provides an alternative to phrase-structure-driven semantic composition based on a fragment of Linear Logic (Girard, 1987). Two recent developments (Kokkonidis, 2007a; Kokkonidis, 2006) concerning the formal foundation of Glue Semantics, its type-system, lead to the latter becoming suitable not only for being an interface between a syntax and a semantics formalism (Glue), but also a variety of syntactic formalisms based on a simple first-order linear type system. TL-LFG (Kokkonidis, 2007b) is one such formalism. It combines G3i (Figure 1), the latest First-Order Glue type system, with a simple mechanism for checking f-structure constraints.

This paper presents the two basic formal components of TL-LFG, the type system used for syntactic analysis and semantic composition, and the regular-expression checker enforcing the constraint equations that ensure that any syntactically unacceptable f-structures are rejected.

2 The standard sentence parsing setup

In TL-LFG, syntactic analysis is not about syntactic trees; it is about semantic composition. The question is not “Given this sentence, what syntactic structure do I map it to?”, but rather “Given this sentence, what is its meaning?”. Syntactic information is gathered and put together (by means of feature unification) in the process of meaning assembly, not for its own sake but for guiding that very process, making the interface of syntax and semantics a very intimate two-way affair.

Type-driven semantic composition is also the approach taken by Type-Logical Categorial Grammar (Morrill, 1994; Moorgat, 1997). However, TLCG does not go all the way in abandoning syntactic categories and using semantically-motivated types directly in describing syntactic structure. TL-LFG, like Dynamic Syntax (Kempson et al., 2001) before it, does exactly that. So, the basic idea when parsing a sentence is that its semantic contributions put together (each used exactly once) should form an expression of type $t$ (truth statement) i.e. the type of the meaning of a
complete clause/sentence. The syntax-semantics interface (SSI) types in TL-LFG typically are the linear counterparts of the corresponding semantic types\textsuperscript{24} with some added syntactic information necessary for driving semantic composition.

The syntactic analysis and semantic composition problem for sentences is: “Given a typing context \( \Gamma \) consisting of the semantic contributions of a word sequence \( \gamma \) what are, if any, the possible expressions \( M \) such that

\[
\Gamma \vdash M : t_f
\]

where

\[
\Gamma = \begin{bmatrix}
0 \ldots N \\
\vdots \\
\end{bmatrix}
\]

and \( N \) is the number of words in \( \gamma \).\textsuperscript{25}

Here is an example. The word sequence \( \gamma \) in (1) is a sentence.\textsuperscript{26} Therefore there will be expressions \( M \) satisfying the above judgement.

(1) 0 Every 1 student 2 loves 3 some 4 book 5

The TL-LFG typing context for (10), \( \Gamma \), consisting of the five semantic atoms is:

(2) every : \( (e_2 \to t_2) \to (e_3 \to t_3) \to t_3 \),

student : \( e_3 \to t_3 \),

loves : \( e_3 \to e_6 \to t_6 \),

some : \( (e_2 \to t_2) \to e_3 \to t_3 \to t_3 \),

book : \( e_5 \to t_5 \).

The typing system of Figure 1 is charged with the task of finding terms \( M \) such that \( \Gamma \vdash M : t_f \). It discovers that the two ways meaning atoms can be combined into a meaning \( M \) for the entire sentence are:

\[
\begin{align*}
(3) & (\text{Reading 1}) \\
& \text{every} \quad (\lambda x.\ \text{book} \ x) \\
& \text{an} \quad (\lambda x.\ \text{an} \ (\lambda y.\ \text{view} \ y)) \\
& \text{loves} \quad (\lambda y.\ \text{loves} \ x \ y) \\
& \text{some} \quad (\lambda x.\ \text{student} \ x) \\
& \text{book} \quad (\lambda x.\ \text{loves} \ x \ y)
\end{align*}
\]

According to the Principle of Compositionality the meaning of the whole should be a function of the meanings of the parts. Replacing the meaning placeholders every, student, etc. with their intended meanings gives a potential meaning for the sentence. Each of the two expressions \( M \) represents the body of a function from the meanings of the semantic atoms to the meaning of the sentence.

This takes care of the semantics side i.e. what is on the left of the colon. Next, the focus will be on what happens on its right.

3 The derivational component

The meaning of each atom contributes to the meaning of the whole, but does not in itself have a role in how it combines with other meanings. It is the syntax-semantics interface types that drive the meaning assembly process.

The type system of Figure 1 is the heart of the formalism. It is essentially the same as the system proposed for Glue by Kokkonidis (2006). Although it serves a different purpose in TL-LFG, virtually all results and intuitions from the Glue Semantics literature carry over to TL-LFG.

The logic of the system is surprisingly simpler than the underlying logics for systems for Type-Logical Categorial Grammars. It is a commutative but also a resource-sensitive logic, so the typing context can be thought of as a multiset, not a set, but not a list either. It only has an introduction and an elimination rule for \( \to \). Finally, it is a first-order logic, but one based on unification rather than quantification.

\textsuperscript{24} Each semantic contribution is meant to be used exactly once. This is guaranteed using a linear but not an intuitionistic type system.

\textsuperscript{25} The vertical ellipses in an attribute-value matrix mean that a number of features have been omitted and they all have fresh variables as their value i.e. are fully underspecified.

\textsuperscript{26} The role of the numbers preceding and following each word will become clear when word spans are discussed. The number preceding a word is the number of words preceding it; the number following it is that plus one.
Unification is represented in terms of an assignment function $\sigma$ mapping variables to values and substitution. Two expressions $f_1$ and $f_2$ unify if and only if there is an assignment function $\sigma$ such that $f_1[\sigma]$ (i.e. $f_1$ with all variables in the domain of $\sigma$ replaced by their $\sigma$-images) equals $f_2[\sigma]$ (i.e. $f_2$ with all variables in the domain of $\sigma$ replaced by their $\sigma$-images). So $e(\alpha)$ and $e(\beta)$ may unify because there can be a function mapping $\alpha$ to 5, but $e(\alpha, 5)$ and $e(6, \alpha)$ don’t because there can not be a function mapping $\alpha$ to both 5 and 6.

The $\to \to$ Elimination rule is different to the one commonly found in the $\lambda$-calculus literature. The intuition here is that a functor such as $\text{loves} : e_s \to e_o \to e_t$ is better seen as a function from two entities to a truth value, rather than as a function from a truth entity to a function from a truth entity to a truth value. Another key difference is the requirement that $F$ be an atom and $T_{N+1}$ a base type. These details ensure the type system only deals with expressions in a particular normal form. Important as they are (they guarantee parsing decidability) they will not concern us here.

What the $\to \to$ Elimination rule states is also simple. If it would be possible to form an expression $E$ of type $T'$ given an additional resource $X:T$, then it is possible to form a function from $X$ to $E$ without it, the new function’s type being $T \to \to T'$.

Kokkonidis (2006) discusses how the system is to be used in Glue Semantics and all the details of that discussion are relevant but with one complication. Whereas in Glue Semantics mere atomic labels are used, in TL-LFG it is entire encoded feature structures that are part of the base types. In our discussion, Greek letters have been used as variables which can be instantiated to f-structures and s, s', o, o', s', s', o', o', f and f' simply stand for f-structures. It just so happens that there is an assignment function, $\sigma$, such that $s[\sigma] = s[\sigma]$, $o[\sigma] = o[\sigma]$, $s'[\sigma] = s'[\sigma]$, $o'[\sigma] = o'[\sigma]$, $f[\sigma] = f[\sigma]$, and $\sigma(\alpha) = \sigma(\beta) = f$ and this allows the derivation to proceed. While the question of why is the equality $\sigma(\alpha) = \sigma(\beta) = f$ part of the requirements for a suitable $\sigma$ is relevant also to Glue Semantics, the equations preceding it are special to TL-LFG and very relevant to the discussion of the syntax-semantics interface in TL-LFG works.

The equality $\sigma(\alpha) = \sigma(\beta) = f$ is needed because quantifiers and generalised quantifiers essentially contribute an entity; therefore, unification or universal quantification over the argument of the truth-type constructor ($t/l \ i.e. \ t$ of arity 1) in the return type of their nuclear scope predicate parameter is required to ensure they only restrict the candidate predicates by the $(e)$ argument type, not the $(t)$ return type (Kokkonidis, 2007b). The equality says that the variable is instantiated to the only sensible option in the example, the f-structure for the entire sentence.

![Figure 1: The TL-LFG Type System](image)
In our previous example, we had a number of pairs of underlined and non-underlined f-structure expressions. What we start with in the target type, f, does not contain much information, but f contains the complete f-structure representation of the sentence. Actually, f[σ] = f[σ], or in other words f and f unify. The same is true for all the remaining pairs of underlined and non-underlined f-structures. This means that, for instance, all agreement requirements are met.

To illustrate the workings of the type system in full, a much simpler example will be used:

\[\lambda x. \text{run}(x) : e\left[\frac{\alpha \cdot \nu}{\text{ORTH orth}}\right][1] \rightarrow o t\left[\frac{\alpha \cdot \nu+1}{\text{ORTH 'run'}} \right] \text{SUBJ [1]}\]

Figure 2: Lexicon entry for ‘run’.

\[\text{[indexicalII]} : e\left[\frac{\nu \cdot \nu+1}{\text{ORTH 'I'}}\right]\]

Figure 3: Lexicon entry for ‘I’

What was underspecified in the target type, is now completely specified (the SPAN, ORTH and SPEC attributes have non-null values, the rest are null).

4 The encoding of f-structures

The type system of TL-LFG is based on a first-order logic. First-order logics are logics about facts about individuals. Constants and variables can be individuals but what allows the encoding of hierarchical f-structures in the type system of TL-LFG (Kokkonidis, 2007c) is the notion of functions from a number of individuals (children f-structures) to an individual (an f-structure containing them). Such an encoding (not the most efficient possible, but arguably the simplest) will be outlined here briefly.

4.1 The encoding of f-structures

In LFG, an f-structure is a partial function from attributes (feature names) to their values. Such a partial function has a corresponding total function that maps any missing attribute to a special value: ⊥ (pronounced null). A total-function f-structure can be represented in TL-LFG with the help of an N-ary function fstr where N is the size of the set of possible f-structure attributes. Each argument position corresponds to an attribute (possibly ⊥ if, in LFG terms, the attribute does not exist in that f-structure).

Given that f-structures found as values of certain attributes will only have a specific subset of
attributes this encoding is particularly wasteful if used as-is in an actual implementation. The simple solution is to have a different function for different types of f-structure (Kokkonidis, 2007c). However, this discussion is beyond the scope of this paper.

4.2 The encoding of spans

The idea of numbering positions within the word sequence borrowed from chart parsing helps deal with word order. Dealing with word order requires syntactic categories, phrase structure rules, c-structure, and a mapping from c-structure to f-structure in LFG, or a split of implication and special modalities in TLCG. Handling word order in TL-LFG requires nothing more than the addition of an attribute SPAN inside of which one finds the attributes START and END with their numeric values. Spans may be written in a special way, on the first line of an f-structure containing them, but they are not a special formal construct.

4.3 Lists

TL-LFG has a very simple concept of a list. A list can either be empty or consist of a head and a tail, the latter being a pair of a value (the head) and a list (the tail). The empty list is simply represented by the null value (┴). The non-empty list is simply an f-structure with two attributes HEAD and TAIL:

\[
\begin{array}{c}
\text{HEAD} \quad \chi \\
\text{TAIL} \quad \tau
\end{array}
\]

For convenience, an alternative notation can be used: \(<\chi | \tau>\). Furthermore, \(<\chi_1, \ldots, \chi_n | \tau>\) can be written instead of \(<\chi_1 | <\chi_2 | \ldots | <\chi_n | \tau>>\) and \(<\chi_1, \ldots, \chi_n>\) instead of \(<\chi_1, \ldots, \chi_n | \bot>\).

LFG has two concepts of a set: the pure set and a weird hybrid structure combining a set and a normal f-structure. Neither of these concepts seems in any way essential. Lists, not a new element of the formalism, but one definable in terms of simple features play a similar role to LFG sets in TL-LFG. Kokkonidis (2007c) offers a brief discussion on this issue.

5 The LFG well-formedness conditions

In addition to whatever constraints a particular grammar may impose on an LFG f-structure, there are three fundamental well-formedness conditions it must adhere to: Completeness, Coherence, and Consistency (Kaplan and Bresnan, 1982; Dalrymple, 2001). In summary, they ensure that each f-structure PREDicate has all its necessary arguments and that each attribute of an f-structure has no more than a single value. In LFG, these conditions, especially Coherence and Completeness, need to be made explicit. In TL-LFG, they follow from the syntax analysis / semantic composition setup. In LFG, it would not be the phrase structure rules that would not permit sentences such as:

(5) *Alex said.
(6) *Basil yawned Catherine.

These sentences violate the Completeness and Coherence requirements respectively. The idea is that the f-structure PREDicate comes with a list of governable grammatical functions (SUBJ, OBJ, OBL, COMP, and XCOMP) corresponding to the predicate’s arguments, and a list of non-governable functions (ADJ, XADJ). Completeness and coherence are both concerned with the first list, that of governable functions.

Kaplan and Bresnan (1982, pages 211-212) define completeness and coherence as follows: An f-structure is locally complete if and only if it contains all the governable grammatical functions that its predicate governs. An f-structure is complete if and only if all its subsidiary f-structures are locally complete. An f-structure is locally coherent if and only if all the governable grammatical functions that it contains are governed by a local predicate. An f-structure is coherent if and only if all its subsidiary f-structures are locally coherent.

So example (5) is incomplete because the main f-structure is missing a grammatical function feature that its verb predicate requires, whereas (6) is incoherent because it has an extraneous OBJ grammatical function feature.

In a Context-Free Grammar (CFG) it is usual to capture similar facts by a proliferation of syntactic categories and rules. A verb\(^29\) in English can be transitive, intransitive, ditransitive, etc. The number of verb subcategorisation frames can be very

---

\(^28\) This is not surprising given that similar results were demonstrated for Glue Semantics.

\(^29\) This discussion of completeness and coherence will concentrate on verbs only because they make an interesting case.
high. There is an argument against the CFG approach that says that once what the subcategorisation frame of a verb is has been specified in the lexicon, it is wasteful to a need to also have a proliferation of phrase structure rules dealing with subcategorisation too.

There is nothing stopping LFG researchers from adopting the CFG approach in writing phrase structure rules. However, following Kaplan and Bresnan (1982) the preferred approach is to try to make LFG phrase structure rules as simple and generic as possible and leave completeness and coherence on one hand and the lexical entries for verbs with their PRED and their list of arguments on the other take care of whether a verb has been given the correct arguments. This is in many ways a more elegant approach than CFG’s.

TL-LFG takes such improvements one step further. In LFG, one need not worry about introducing multiple new syntactic categories for verbs that take different numbers and types of arguments; in TL-LFG one does not have to worry about introducing syntactic categories at all. In LFG, one need not write a bunch of different phrase structure rules for verb phrases headed by different subcategories of verb; in TL-LFG, one need not write phrase structure rules at all. In LFG, all that is needed to ensure that subcategorisation checks are made is a mechanism for checking the completeness and coherence conditions in f-structure; in TL-LFG completeness and coherence are guaranteed without further stipulation. Overall, TL-LFG’s approach is in many ways simpler and more elegant than LFG’s.

The Consistency (or Uniqueness) Condition guarantees that the relation between attributes and values in LFG is a function: in a given f-structure, a particular attribute may have at most one value (Kaplan and Bresnan, 1982). This is also automatically guaranteed by the TL-LFG type system without need for a dedicated well-formedness condition.

So, the three fundamental LFG well-formedness conditions come for free with TL-LFG. This can be seen as evidence for an argument that TL-LFG captures theoretical intuitions more directly that LFG.

6 Grammatical Constraints in TL-LFG

Defining equations, the single most important kind of f-structure constraint in LFG, is absent in TL-LFG as their job is taken over by the type specifications.

Constraining equations, on the other hand, are available. Given σ, the minimum assignment function needed to allow the derivation to work, a variable x in its domain and a value v, the constraining equation x = v succeeds if and only if there is an assignment function σ’ such that

1. σ’ ⊇ σ and
2. (σ(α))[σ’] = v[σ’].

If, for example, σ = {(α, p(β))}, then the constraining equation α =c p(1) is satisfied; this is because choosing σ’ = {(α, p(β)), (β, 1)} satisfies both the condition that σ’ is a superset of σ and (σ(α))[σ’] = (p(β))[σ’] = p[σ’] = p(β) = p(1) = p[σ’] = p(1)[σ’].

Much more interesting constraints can be expressed using functional uncertainty. Functional uncertainty has been successfully used in analyses of long-distance dependencies in LFG. In TL-LFG, it is the type system that deals with long-distance dependencies, and it does so using variables (fully underspecified f-structures). Checking equations possibly involving functional uncertainty ensure that the variable used has not accidentally been instantiated to the wrong f-structure. The verification problem of TL-LFG is much simpler than the satisfiability problem of LFG (Kaplan and Maxwell, 1988) and it involves little more than a regular-expression recogniser working on the structure of f-structures.

In LFG, there are two kinds of functional uncertainty checks:

32 While some constraints can be checked during the syntactic analysis / semantic composition derivation, it may be the case that some constraint checked too early will provide the wrong answer due to the fact that the data it is applied to is not yet complete. It will be assumed for simplicity that all f-structure grammatical constraints described below are performed after the syntactic analysis / composition derivation.
1. Outside-in functional uncertainty, first introduced by Kaplan et al. (1987) and Kaplan and Zaenen (1989), is used to define constraints on more deeply embedded structures.

2. Inside-out functional uncertainty, first introduced by Kaplan (1988), is used to define constraints on enclosing structures.

In TL-LFG there is only one kind of functional uncertainty, but that does not prevent TL-LFG from expressing the same constraints. Here are the formal definitions:

An f-structure expression can be either an f-structure f, or a choice between two f-structure expressions F1 and F2 (F1 | F2), or an f-structure expression F followed by a path expression P (F.P).

A path expression can be either an attribute a, an attribute a with an off-path constraint o (a :: o), a sequence of two path expressions P1 and P2 (P1 | P2), a choice of two path expressions P1 and P2 (P1 | P2), an optional path expression P (P*), a path expression P that can be followed zero or more times (P), or a path expression P that can be followed at least once but possibly more than once (P*).

The relation→ links f-expressions with f-structures they may reduce to. Let f, f', and f1 be f-structures, l a list f-structure, a an attribute, o an off-path constraint, F, F', F1, and F2 f-expressions, and P, P1, and P2. The relation → between f-expressions and f-structures is the smallest relation such that:

- If f1 = f, then f1 = f.
- If F1 → f1 and (F2 | F1) → f1.
- If F → f and f.P → f; then F.P → f.
- If a is defined for f then f.a → f.
- If o[s34 f'], then f.a::o → f'.
- If f.P1 → f' and f'.P2 → f'', then f.(P1 | P2) → f''.
- If f.(P1 | P2) → f1 and f.(P2 | P1) → f1.
- If f.P → f' then f.P' → f and f.P' → f.
- If f.P → f' then f.P' → f' and f.P' → f'.

- If F.P' → f', then f.P' → f and f.P' → f.
- If l( TAIL HEAD) → f (i.e., f is a member of the list l) then l(MEMBER → f).

Given σ, the minimum assignment function needed to allow the derivation to work, a check constraint of the form F1 ≡ F2, where F1 and F2 are f-structure expressions, is satisfied if and only if F1 → f1 and F2 → f2, and there is an assignment function σ' such that

1. σ' ⊇ σ and
2. f1 [ σ'] = f2 [ σ']

Inequalities are simply the negation of the corresponding equalities.

To check if some f-expression F is a σ-unassigned variable (i.e., not in the domain of σ), one can use the following trick: ¬(F ≡ 0 ^ F ≡ 1). The syntactic sugar for this is: var(F).

Since there is a value in TL-LFG given to empty attributes, ⊥, the equivalent of LFG negative/positive existential constraint can simply be expressed as a checking equality/inequality constraint involving that value.\(^{35}\)

The entry for the relative pronoun ‘who’, with many details left out\(^{36}\), will serve as an example of how functional uncertainty checks are used in TL-LFG:

\[ \text{who}: (e_{a \rightarrow o \ t_{2}}) \ldots (e_{e \rightarrow o \ t_{2}}) \ldots (e_{e \rightarrow o \ t_{2}}) \]

where

\[
\begin{align*}
\text{who} & = \lambda c.\lambda n.\lambda \chi.\text{N}(\chi) \land C(\chi) \land \text{person}(\chi) \\
& \quad \quad \text{TOPIC} \quad \text{PRONTYPE} \quad \text{rel} \quad [1] \\
& \quad \quad \text{RELPRO} \quad [1] \\
& \quad \quad \vdots \\
& \quad \quad \vdots \\
\end{align*}
\]

Given a clause (t_{2}) missing an entity (e_{a}), and a noun (e_{e \rightarrow o \ t_{2}}) who returns a modified noun

\(^{33}\) Trivially true for the described encoding but not others.

\(^{34}\) The dollar sign is used in TL-LFG off-path constraints instead of the LFG left arrow to avoid confusion. It represents the value of attribute a.

\(^{35}\) It is advisable to combine ‘existential’ constraints with constraints checking if any of the parties is a variable; otherwise, the result may be unexpected.

\(^{36}\) Dalrymple (2001) presents a detailed LFG account on which the current one is loosely based.
As $\alpha$ is a variable it may unify with any entity f-structure in the typing context, resourcesensitivity permitting. The constraining equation disallows inappropriate f-structure unifications.

7 Conclusions

TL-LFG shows an alternative way of approaching syntax-semantics challenges different to that of LFG. This paper has covered the two key formal components of TL-LFG, the type system and the constraining equations checker. If less is more then TL-LFG has achieved its design objectives. However, only more linguistic analyses using it will reveal if that economy in formal devices leads to a formalism that is better, in any practical sense of the word, than LFG combined with Glue. Kokkonidis (2007d) takes a first step in that direction, but more will be needed.

Acknowledgements

I would like to thank Mary Dalrymple and the two anonymous referees for commenting on an earlier version of this paper. This work has been supported by AHRC grant 2005/118667.

References


Document Editing through Speech

Dawn Borg Costanzi
University of Malta
dbor015@um.edu.mt

Abstract
As progress is continuously made within the computerised world, more attention is given to the accessibility of computer applications. We aim to lift the limitations imposed on the dexterity impaired by the dependence on manual devices. We look into a number of language issues, to create systems which are capable of accepting intuitive, spoken commands and interpreting them to perform appropriate tasks in the particular domains of text and table editing. Through the consideration of natural language in the development of speech-controlled editing applications, this paper gives useful insight into one particular area of accessible computing.

1 Introduction
This project is motivated by the realisation of the extent of dependence of computer users on manual control, which evidently causes problems for those with dexterity impairments. Ideally, all computer applications should be made accessible to people with different kind of disabilities, especially since computers have increasingly become a way of life.

This project takes a small step in the direction of accessible computer applications, incorporating areas of natural language processing to tackle speech control of document processing. It attempts to deliver a system which is capable of working with intuitive, natural-language commands.

1.1 Aims and Objectives
The main aim is to improve accessibility of a computerised field which is widely used, by providing natural language speech control for a limited range of visually-based tasks.

In doing so, we can gain a better understanding of the language issues associated with the use of natural and flexible commands. We also determine the possibility of developing solutions using readily-available components and consider the underlying framework necessary to extend the approach to various different applications.

2 Background
2.1 Choice of Application Domains
The main objective is to create software for speech control over tasks of varying difficulty, with the general criteria of being a visual task for which there is some social need. Different domains were considered, and tests carried out to gather the typical language used to control each of them. The two domains implemented are those of text and table editing.

2.2 Text Editing
Text is considered in its most basic form without considering any formatting. The idea of providing speech control for this domain stems from the fact that text is used throughout a wide range of sophisticated applications and solving such a problem would satisfy a widespread need, whilst paving the way for further domains to be tackled.

2.3 Table Editing
Table editing builds on text editing by adding a structured layout and possibly formatting. Table manipulation purposes include data display and web design, which introduces a wider range of tasks. The visual aspect of table editing is stronger than that of text editing as it includes factors such as positions, colours and sizes. Analysis of typical commands shows that this introduces issues which are not encountered with text editing, and therefore provides a higher level of complexity.
3 Design and Implementation

The system essentially make up the first part of a dialogue system: a spoken phrase is accepted, in this case a command, and interpreted to perform some processing. However, this system is not expected to produce spoken responses as a dialogue system might, but it is required to produce output in the form of an update to the text or table as instructed by the user. The figure below shows this general architecture.

3.1 Speech Recognition

The speech recognition component converts speech signals into system commands through full language specification of valid commands, to extract the domain-relevant information contained within the utterance. This component could not be designed and implemented, so a ready-to-use speech development kit was sought out.

The engine must give accurate results, whilst making the recognition details accessible. It must handle command-and-control functionality by referring to a grammar and lexicon, to accept a specific range of commands rather than any dictation. The possibility to extend the recognition model by adding words, and tune the recognition model by encoding different pronunciations is also necessary. This is important since we use British English and most toolkits use American English.

Also, this unit must integrate smoothly with the rest of the system, and be accessible at a high-level within the development environment.

The Microsoft Speech SDK satisfies these requirements well. Having been used in many other projects, the recognition engine is understood to perform properly and present results in a favourable format. Command-and-control support is offered and alteration and augmentation of the lexicon is possible. Interfaces are also included for smooth combination with applications developed in the Microsoft .NET environment. These features help ensure that the speech recognition component is right for this project and that its use will not hinder development but rather make completion of the overall task easier.

3.2 Front – End Editing Applications

The front-end management component interfaces the editing applications with the processing of the user instructions. A model of the application is stored for updates to be applied according to the actions triggered by given commands.

The text and table editing application domains require separate front-ends in order to cater for their differences in functionality provided. 

Text Editing

A text processing application from scratch, due to the simplicity of the front-end required.

It was decided that only the functions directly related to editing text would be provided with a speech interface. This lets the domain remain restricted enough for proper study, whilst eliminating details necessary to carry out externally based functions, such as opening or saving files.

The text editing tasks are typical operations such as copying, pasting, finding and replacing text, and so on. Functions are added to emulate the most fundamental of manual operations and perform navigation and selection.

![Diagram](image)

Figure 1. General Architecture
Table Editing
For table editing the complexity of creating a front-end environment may be avoided by taking advantage of API’s for table manipulation.

Microsoft Office API’s, especially those related to word processing, were looked into. Primary Interop Assemblies allow for Word’s Object Model to be utilised through Visual Studio, to carry out table editing tasks in .NET applications.

Table editing builds upon text editing, so previous functionality is offered and additionally includes tasks necessary to create data and web design tables. A set of operations were selected for speech control, providing enough support to create the type of tables required and demonstrate the typical language. These tasks allow the user to create tables, insert rows and columns, merge cells, arrange widths, height and colours, and so on. Basic text editing is enhanced by operations which deal with the formatting, mainly modification of fonts. The navigation and selection commands are still catered for, but they are modified in accordance with the table editing domain.

3.3 Language Understanding
The language understanding component interprets recognition information to convert it into the programmatic command for the user instruction. This component includes mechanisms to analyse recognition information, resolve referring expressions, handle contextual information and create a front-end executable command.

Grammar and Lexicon
The grammar and lexicon are used by the speech component, but are important for the language understanding as they ensure recognition of acceptable structures. The grammar is semantically augmented so syntactic parsing may be carried out by the speech unit and the understanding process may begin from a semantic representation.

To design a grammar which satisfies intuitive language, studies were carried out to observe user language. User utterances were gathered and, through generalisation, a single rule was composed for each operation. Meaning representations, in the form of properties and values, were devised so the necessary semantics are available with recognition.

Once the entire recognition grammar was designed, the associated lexicon was built, by creating a list of all the words within the grammar rules, along with their phonetic pronunciations.

Recognition Information Analysrer
The recognition information analyser manages the entire language understanding process.

This receives information from the speech recognition component. It identifies the main grammar rule which fired, to determine which operation has been requested and the sequence of tasks which must be followed. The semantic properties are then used to perform context handling and possibly reference resolution.

The analyser interacts with the front-end management component through some programmatic command, to carry out the required operation. It may process further information to ensure that the context reflects the updates made.

Context Handler
The types of contextual reference permitted are:
- the use of the pronoun ‘it’
- the use of ‘the’ (eg: ‘the line’)
- the use of ‘this’ (eg: ‘this cell’)
- the use of one-anaphora (eg: ‘the last one’)

To interpret such phrases, a model of the current focus within the application is kept, reflecting the physical details of domain entities. This is built on the stack model of Grosz and Sidner (1986).

For the text editing domain, the focus spaces stored include the current paragraph, line, word and character. For table editing, physical locations for the row and column are added. This reflects the way logical and graphical positions are referred to.

It is also important to store discourse focus, for a complete representation of the current situation. Thus, the focus space also retains the last mentioned structures.

As suggested, focus spaces are added whenever a shift in attention occurs. To determine shifts, the current focus space is compared to the command’s semantic information. If the new information adds detail to the focus space, it is retained and the new information augmented, otherwise a new focus space is added. Focus spaces are not removed from the top of the focus stack, but retained, since the language is a set of individual commands.

Reference Resolution
The resolution component is responsible for finding the referent of descriptive reference, such as ‘the widest green cell’. In the text domain, resolution of descriptions is trivial and requires only a comparison of some intrinsic property of the text. Thus, it was decided only to allow descriptive referring expressions in the table editing domain.
The descriptive properties allowed are:
- typological (eg: ‘row’, ‘cell’)
- intrinsic, describing colour, size and shape (eg: ‘wide’, ‘big’, ‘blue’)
- relational, describing spatial position (eg: ‘topmost’, ‘in the middle’)

The reference resolution component uses the differential model proposed by Pitel and Sansonnet (2003). Each property requires an extractor, composed of a comparison function to order the entities by some property, an exclusion function to eliminate impossible candidates and a preference function to choose the most likely referents.

The order in which extractors are applied depends on context-sensitivity. Thus, the items are first obtained according to their type, and then extractors are applied as follows: colour, position, shape and size. Applying the model in this way ensures that a single referent is found, which satisfies all the descriptive requirements.

4 Evaluation and Results

Evaluation of the system was carried out in a way similar to that recommended for dialogue systems by McTear (2004).

The speech recognition system is the only one with which users can interact and the only one evaluated individually. Users repeated a list of commands to the recognition engine, to draw up the quantitative measures of Word, Sentence and Concept Accuracy.

Text and table editing tasks were set to analyse the success of the entire system. User interaction was monitored, like the correctly completed sub-tasks, system errors, appropriateness of actions taken and the amount of rephrasing required.

Qualitative evaluation was also performed, as users rated the ease of use of the system, whether it understood the user, took appropriate actions and responded in good time, and the way it coped with errors. They were also asked whether they would prefer using such a system rather than their ordinary manual one.

It was found that the speech recognition engine adapts to voice characteristics like pitch and accent, and gave very good results for the developer who had been using it for months. Since the processing of semantic properties is unambiguous, correctly recognised sentences are interpreted properly and lead to the correct operation being performed.

Illegitimate commands were excluded by the grammar, so the few errors met were often a result of incorrect recognition of incorrect commands, which together may generate mistaken instructions.

Natural language flexibility is the system’s greatest advantage. It allows for rephrasing, to overcome the problem of incorrect recognition, and reduces the need for full knowledge of the command set, as instructions may be formed intuitively. Another useful system feature is its handling of contextual information, so that repetition of information unnecessary. Users were also pleased that separate operations could be carried out by one command.

Users had mixed feelings on whether they would choose a speech-controlled system over a manual one. This proves that computer users are highly dependent on manual devices, but does not give any indication of what users may choose to do if they are not able to use manual control.

On the whole, user satisfaction was sufficient to show that the speech-controlled systems function well, and with improvements could provide a good level of accessibility in the associated domains.

References


Statistical Machine Translation for Mobile Devices

Alexandru Ceausu
Research Institute for Artificial Intelligence, Romanian Academy
aceausu@racai.ro

Abstract

The limited resources (storage capacity and processing power) of Pocket PCs, Smart Phones, etc., are severe impediments in deploying complex NLP applications. Statistical Machine Translation (SMT) systems require computing resources that sometimes are beyond the standard specifications of desktop computers. The paper describes a distributed architecture for a SMT system to bring machine translation to the mobile devices using XML Web Services. The system uses a phrase-based decoder and it employs already proven techniques for building the language and translation models.

1 Introduction

A statistical machine translation system has three main components: a language model (LM), a translation model (TM), and a decoder. For a system that translates from English to Romanian, LM gives an a-priori probability $P(s)$ where $s$ is a sequence of words in Romanian. The translation model gives a conditional probability $P(t|s)$ that is the translation probability. The models are automatically acquired using monolingual corpora (for the LM) and multilingual corpora (for the TM). The task of the decoder is to find the Romanian word sequence that fits the best the English word sequence maximizing the product of the conditional and the a-priori probability from the translation and language models.

Due to the computational requirements of a decoder (for storing the language and translation models and for computing the best sequence for translation) we chose to deploy it as a web service.

Mobile devices are most often not standalone solutions since they need to communicate with other systems. We chose XML Web Services as our data access strategy. Using markup standards as XML and HTTP, XML Web Services can publicly provide different functionalities. In this framework messages are exchanged using SOAP (a markup specification similar to HTTP) and the behavior of the XML Web Service is described using WSDL (Web Service Definition Language). WSDL layers provide additional information over the XML Schema definitions that describe the actual messages.

We tested with Koehn's (2003) phrase-based decoder using a table of phrase translation equivalents generated from our translation model, paired with a language model build with SRILM (Stolke, 2002).

The pairs of languages of the translation system are Romanian-English and English-Romanian. To make the task simpler we chose to implement a domain-specific translation system.

2 Translation model

The translation model train data is based on a collection of documents from the Romanian-English language pair of the JRC-Acquis corpus (Steinerberger et al, 2006). The documents in the Romanian-English language pair (6256 documents) were sentence aligned using the SVM-aligner (Ceaușu et al, 2006).

The training of the phrase-based translation model employs a word alignment stage. For word alignment we used a binary classifier library package - LIBSVM (Fan et al, 2005). Trained on just 10000 correctly aligned word pairs, the application achieves 92% precision and 71% recall on a test set of 4000 correctly aligned word pairs.
2.1 Feature selection

From a given parallel corpus one can compute a series of properties for each of its aligned pairs. An aligned pair becomes a structured object which can be manipulated in various ways, independent of the parallel corpus (or even of the sentences or lexical tokens of the pair) from which it was extracted. We call this procedure alignment reification (Tufiş et al 2005, 2006).

In the process of features selection, any word pair can be characterized by a collection of scores for each feature. Therefore, the alignment problem can be reduced to a two-class classification task: discriminating between positive and negative examples. Using this procedure we can transform the process of word alignment in a classificatory task.

The classifier we employed - LIBSVM (Fan 2005) - uses Support Vector Machines (Vapnik 1995) for discriminating between positive and negative examples.

TRANSLATION EQUIVALENTS. This feature may be used for two types of pre-processed data: lemmatized or non-lemmatized input. We use GIZA++ (Och and Ney 2003) to build translation probability lists for either lemmas or the occurrence forms of the bitext. Irrespective of the lemmatization option, the considered token for the translation model build by GIZA++ is the respective lexical item (lemma or wordform) trailed by its POS tag (eg. plane_N, plane_V plane_A). In this way we avoid data sparseness and filter noisy data. A further way of removing the noise created by GIZA++ is to filter out all the translation pairs below LL-threshold. We made various experiments and empirically set the value of this threshold to 6. All the probability losses by this filtering were redistributed proportionally to their initial probabilities to the surviving translation equivalence candidates.

LOCALITY is a feature that estimates the degree to which the links are sticking together.

The value of the weak locality feature is derived from the already existing alignments in a window of N tokens centred on the focused token. The window size is variable, proportional to the sentence length. If in the window there exist k linked tokens and the indexes of their links are <i_1, j_1>, ...<i_k, j_k> then the locality feature of the new link <i_{k+1}, j_{k+1}> is defined by the equation below:

\[
LOC = \min(1, \frac{1}{k} \sum_{m=1}^{k} |i_{k+1} - i_m|)
\]

OBLIQUENESS. Each token in both sides of a bitext is characterized by a position index, computed as the ratio between the relative position in the sentence and the length of the sentence. The absolute value of the difference between tokens’ position indexes, subtracted from 1, gives the link’s “obliqueness”.

\[
OBL(SW_i, TW_j) = 1 - \frac{i}{\text{length}(\text{Sent}_i)} - \frac{j}{\text{length}(\text{Sent}_T)}
\]

COGNATES. The similarity measure, COGN(T_s, T_T), is implemented as a Levenstein metric. Using the COGN test as a filtering device is a heuristic based on the cognate conjecture which says that when the two tokens of a translation pair are orthographically similar, they are very likely to have similar meanings (i.e. they are cognates). The actual implementation of the COGN test includes a language-dependent normalization step, which strips some suffixes, discards the diacritics, reduces some consonant doubling, etc.

PART-OF-SPEECH AFFINITY. In the process of translation, the words tend to be translated by words of the same part-of-speech. When this is not the case, the different syntactic categories are not arbitrary chosen. The part of speech affinity can be easily computed from a gold standard alignment.

2.2 Phrase translation equivalents

Traditionally, phrases are taken to be syntactic constituents of a sentence. They are units of a sentence that can be used to generate other sentences of the language and this is the strategy employed in phrase-based translation: instead of generating translation of individual words in the source language, generate translations of the phrases and assemble the final translation by a permutation of these (see (Koehn et al., 2003) or (Yamada & Knight, 2001)).

In a generative framework, a phrase is an ordered list of adjacent words such that they are all leaves of a syntactic tree that is contained by the syntactic analysis of the sentence. Within its dependency framework, Mel’cuk (1988) puts forward another view of the syntactic phrase: a phrase is a
syntactic tree rooted at the phrase’s head. The tree branches are the surface syntactic dependency relations that are established between the word-forms of the sentence. Evaluating the two views of a syntactic phrase, we find that the dependency formulation has one advantage: the syntactic phrase does not require the adjacency of the word-forms of the sentence, thus also allowing for meaningful discontinuous syntactic units.

Figure 1: Phrase translation equivalents identified using dependency structures

Our goal is to align phrases made out of word-forms that are not necessary adjacent. This approach builds on easy-to-obtain pseudo syntactic structures, based on which produces the phrase alignment. Oriented linkages (if correct) coupled with good translation lexicons are sufficient for achieving a good phrase alignment (Ion & Barbu Mititelu 2006).

3 Language model

Markov models (or chains) (Rabiner 1989) are mathematical models that are very well suited to the problem of ‘guessing’ the next object in a sequence of objects based on the objects that have already appeared. This kind of modeling can answer the problem of which word is most likely to follow in a given sequence of words. For many systems, it seems reasonable to assume that all we need to predict the values of future random variables is the value of the present random variable, and we do not need to know the values of all the past random variables in the sequence. The order of the Markov model determines the amount of history used to predict the next element. That is why models of higher orders perform better. However, the costs (computational power and storing space) of using higher order models grow exponentially.

To train a Romanian language model we used 11228 Romanian documents from the JRC-Acquis corpus. The documents have 3935021 tokens of which 60% are content words. To improve the language model we added to the training data the Romanian corpus AGENDA (a collection of articles from the newspaper Agenda of Timisoara). This corpus consists of 8408185 tokens out of which 55% are content words.

For English, we used the language model built from the EUROPARL corpus (Koehn, 2005).

4 Decoder

The phrase-based decoders are more adequate to be deployed on consumer-like computers because they require less computational resources (as opposed to the more complex decoders based on syntactically motivated translations). Koehn's (2003) experiments showed that it is possible to achieve a satisfactory level of accuracy with relative simple means.

In the process of decoding, the target sentence is split into a sequence of distinct phrases. The simplest assumption is that each phrase is translated into exactly one phrase in the source language. The phrases in the source language can be reordered based on a distortion coefficient computed from the parallel corpus.

The decoding service runs using two different machines: one is a Linux box where we deployed the PHARAOH beam search decoder (Koehn 2003) and the second one is a Windows 2003 Server handling client requests and authentication. The decoder and the server are interconnected through a private channel using the HTTP protocol. The server publicly exposes a web service that can be used by a mobile device client. The client hosted on the mobile device handle only the tokenization of the input text; the translation is made through the web service. We implemented tokenization on the mobile client to allow on-line translation services.

5 Conclusions and future work

We described the architecture of a statistical translation system with a distributed architecture
tailored for mobile devices. We developed all the tools needed for building a phrase-based translation model and for building the language model.

Due to the fact that Romanian has a very complex morphology, the translation model of Romanian-English is considerably better than the translation model of English-Romanian. We didn’t have a benchmark for the BLEU score, but we can estimate that the accuracy of the translations made with the Romanian-English model is equivalent to the current baseline in statistical machine translation.

References


Abstract

In this paper, we investigate the problem of automatically predicting topic boundaries in texts. Three different unsupervised approaches to lexical cohesion-based text segmentation are presented and prior work is extended in two ways. We first propose a solution based on the n-gram similarity between consecutive paragraphs and then we introduce a voting system that we hope to improve the segmentation results.

1 Introduction

Text segmentation can be defined as the automatic identification of boundaries between distinct topics in a stream of text. Interest in automatic text segmentation has increased over the last years, with applications ranging from information retrieval to text summarization, and to story segmentation of video feeds. The importance and relevance of this task should not be underestimated, as good structural organization of text is a prerequisite for many important tasks that deal with the management and presentation of data [4]. Researchers have explored the use of document segmentation to improve automated summarization since the aim of this task is to identify subtopics in a document and then generate a summary from this information. Text segmentation issues are also important when responding to a user’s query in an information retrieval task, where users are not given large quantities of text but only a few short passages. Most recently, a great deal of interest has arisen in using automatic segmentation for the detection of topic and story boundaries in news feeds.

The motivation of this research is to investigate the usefulness of a voting system for text segmentation. There are presented three different solutions for this task and the results are used as inputs for the voting system.

The remainder of this paper is organized as follows: Section 2 presents the previous work in the field of Topic Detection and Tracking. Section 3 presents possible solutions regarding Topic Segmentation. Some final remarks and mention of future work are presented in Section 4.

2 Previous Work

Topic Detection and Tracking was initially sponsored by DARPA in the Translingual Information Detection, Extraction, and Summarization (TIDES) program. This was the first attempt to use topics and addressed problems like (1) topics detection, (2) segmentation and (3) tracking.

Segmentation is the task of segmenting a continuous stream of text into topics - that means correctly locating the boundaries between adjacent topics. The solutions for this problem consist in the usage of Hidden Markov Models, feature extraction and Information Retrieval technologies [1]. Based on the results provided by this first study, a lot of research has been done in the field.

Hearst, in [3], proposes a segmentation system called TextTiling based on text cohesion. She stated that a text is not made up of random words or phrases but that the sentences are connected to each other in two ways: cohesion and coherence. Since texts that contain high numbers of semantically related words (so, there is a high cohesion in the text) generally represent a single topic, she considered that the place where the text cohesion is low represent a good indication of a topic change, therefore a good place to segment the text. Later on, in [5], the authors present different types of text cohesion that can be found in texts and they sug-
gest that, by using them, the results are better than the ones obtained by the use of the TextTilling system. They built a system called SeLeCT, which was based on a lexical chaining that used the types of text cohesion they previously identified. The results obtained using the SeLeCT system in order to segment the text are presented and compared with the ones obtained through the TextTilling.

A different approach that uses the Hidden Markov Models is presented in [2]. The authors criticize the “bag-of-word” approach that is used in the segmentation based on text cohesion, saying that the order of the words in a text is very important and without considering it, “the accurate meaning of language cannot be exactly captured by word co-occurrences only“. They suggest that the collocation should be used in order to have more information about the context of each word and argue that by underlining the obtained results, which are better than the ones that result from using the segmentation based on text cohesion.

3 Our solutions

In this section, a solution for the topic segmentation task is described. Three different methods will be presented, methods that are considered suitable for this task, and a voting system, that it is possible to improve the results obtained by the three methods, will be introduced. The application architecture is presented in Figure 1.

Every module from one to three will implement an unsupervised lexical cohesion-based solution. They will be used in order to compare the results obtained by different methods based on lexical cohesion and to see if they can be improved by combining their results using a voting system. It is considered that the text is structured in paragraphs, and that every paragraph belongs to one topic. This is somehow logical, since a topic shift cannot occur in the paragraph that contains the previous topic.

3.1 Tagging Module

In this module the raw text is preprocessed using LPost, which is a probabilistic parts-of-speech tagger built by Jimmy Lin from the University of Maryland [6].

LPost is a Perl program that reads text and for each token in the text returns the part-of-speech (e.g. noun, verb, punctuation, etc). It is based on Benjamin Han's ePost package and it uses a tag-set that is a variant of the Brown/Penn-style tag-sets. Because it uses statistical methods, it does make mistakes, but it is fairly robust and tags texts with good accuracy (96% - 97%).

3.2 Module 1

Since it is considered that every paragraph belongs to one topic, the limits between paragraphs could represent topic shifts; therefore, they are considered candidate topic limits. The first method inspects these candidate limits in order to decide which ones are real limits. Whenever a new candidate topic limit is encountered, it will be evaluated to see if it is a real topic shift. If it is not, it will be considered that the current paragraph is part of the last topic, so this candidate topic limit is ignored and the analysis continues with the next paragraph.

In order to evaluate if a candidate topic limit is a real one, the lexical cohesion assumption is used. Therefore, it is considered that two texts that present a high cohesion – they have the same vocabulary and collocations – are part of the same topic and when a significant drop in cohesion is encountered, it means that a topic shift was found. In conclusion, the text from each side of the limit is separately considered. For each side, all the nouns provided by the Tagger Module are extracted in order to construct a unigram model, then a bigram model, and, if necessary, a trigram model. After that, the models obtained are compared
against each other, and if the similarity is higher than a threshold, than the two texts are considered as part of the same topic. If not, the two texts are part of different topics and the limit between them will be marked. If the decision is that the limit is not real, it means that the topic is spread over more paragraphs and this will be taken into consideration in the next comparison. At the next paragraph limit, when the (unigram/ bigram/ trigram) models for the current topic are computed, not only the last paragraph will be considered, but all the paragraphs that are part of that topic. The threshold will be determined empirically, through experiments. When the similarity score is calculated, the importance of each model in the final result can be modified by scaling them with some scaling factors. For example, the similarity score can be computed as follows:

\[
\text{Similarity} = a \ast \text{Unigram\_similarity} + b \ast \text{Bigram\_similarity} + c \ast \text{Trigram\_similarity}
\]

where a, b, and c are also empirically determined so that \(a + b + c = 1\). The Unigram/ Bigram/ Trigram similarity represents the percentage of the common Unigrams/ Bigrams/ Trigrams from the two texts and it can be computed using the Cosine or the Hellinger distance.

### 3.3 Module 2

The solution presented in this section is based on lexical chains and a clustering algorithm. This solution has been presented in [5]. After tagging, the positions of the important tokens are taken into consideration for grouping in clusters. The resulted clusters indicate candidate topic limits. These limits are evaluated using an algorithm similar to the one presented in the previous section.

An important part of this method is how the lexical chains are constructed. The aim of these chains is to find relations between different tokens based on lexical cohesion measures [5]. The authors identified different types of lexical cohesion:

- repetition (when the same word is repeated later in the text);
- repetition through synonymy (when different words that share the same meaning are found in the text);
- word association through part-whole / whole-part relationships (when these kind of relationships exist between two words - e.g. a library collection is made up of books);

These types of lexical cohesion will be used to group the tokens into lexical chains. In order to identify the synonymy and part-whole/whole-part relationships, we will use the WordNet database, which groups nouns, verbs, adjectives and adverbs into “sets of cognitive synonyms (synsets), each expressing a distinct concept” [7].

The clustering algorithm takes the first token and creates the first lexical chain. Subsequent tokens are compared with the tokens from the existing lexical chains using the Cosine or the Hellinger metrics. If the similarity with every chain is smaller than a threshold, than the token is not part of any chain and it is used as a seed for a new chain. If the similarity between a token and a chain is bigger than the threshold, it can be added to the chain. However, before adding it, the similarity that would result inside the chain after the addition will be computed. This is done in order to avoid the addition of tokens that are used with a different meaning than the one of the tokens in the chain (e.g. the addition of gas to a chain related to air when gas refers to petroleum). If the new similarity is greater than a threshold, then the token will be added to the chain. Otherwise, it will be used as a seed for a new lexical chain. In this new chain, information about the new meaning of the token will be added (the other meaning than the one that has been investigated in the case of the words with only two meanings, or the investigated meaning introduced as an antonym in the case of the words with multiple meaning). Since it is possible that a token is inserted in multiple chains, the ones that have the highest similarity score will be identified and the token will be inserted in the chain that is found first. Therefore, the order in which the chains are compared with the new token is very important. The most recently updated chains are used first to the comparison with the new tokens.

After inserting all the tokens into chains, the topic shifts can be identified, assuming that a high density of chain starts and ends means that the topic has changed. When such a high density is encountered, the place is marked as being a candidate topic limit and then is verified using the algorithm described in the previous section.

### 3.4 Module 3

The solution presented in this section is similar up to one point to the one presented in the previous section. It is a two steps method: in the first step,
the lexical chains are built and in the second one, these chains are used in order to identify the topic shifts. This method implies two scans of the document: the first one is used to build the lexical chains and the second one is used to identify the topic shifts.

As a first step, the lexical chaining method and the clustering algorithm presented above are used in order to create the lexical chains. The difference between the two methods comes after these chains are constructed. Instead of identifying where a high density of chain starts and ends was encountered, we will proceed as described below.

As a second step, numbers will be added to the lexical chains (c₁, c₂, c₃ and so on). Then all the tokens from the text will be replaced with the number of the chain that they are part of. It may be considered that every lexical chain represents one topic because of the way they were built. After that, for each paragraph in the text, we will consider that it belongs to the topic that has the majority number of words in that paragraph. If it is found that two consecutive paragraphs belong to different topics, the place is marked as being a topic shift. By using this method, it will also be possible to specify the domain of the topic because of the words from the lexical chain, words that gave the background of the topic.

3.5 Voting Module

In this module, an attempt to combine the results obtained from the Modules 1 to 3 will be made in order to see if any increase in the final results can be achieved by doing that. An examination will be made to see if the methods are complementary and if some scaling factors could be used in order to obtain a higher percentage of correct results. The scaling factors will be empirically determined.

It is also intended to see if there are methods that give the same results in order to avoid giving them too much importance in the final results, by choosing the scaling factors accordingly. Interest in identifying the causes that make two different methods act identically is present.

3.6 Statistics Module

In this module, we will compare the results obtained by every method and by the Voting system against the Golden Standard using the WindowDiff, a new evaluation metric proposed by Pezvner and Hearst [4]. The results obtained by every system separately will also be compared with the ones obtained by the voting system.

Statistics will be added with the scaling factors from the first method and how they influence the final results will be shown. The results obtained using the Cosine metrics will be compared against the ones obtained using the Hellinger metrics.

Other statistics will be related to the different scaling factors from the Voting Module.

4 Summary and Further Work

In this paper we proposed a voting system for text segmentation hoping that the suggested methods will be complementary and they will be able to correct one other’s mistakes. Every used method is an unsupervised cohesion-based solution.

In the future, we plan to use the same voting system to integrate some semi supervised techniques used for text segmentation. We already thought of solutions like Latent Dirichlet Allocation and solutions based on Hidden Markov Model.

References


4 Discourse Processing
Textual Entailment

Adrian Iftene
Research Institute for Artificial Intelligence, Romanian Academy
„Al. I. Cuza“ University, Faculty of Computer Science, Iasi, Romania
adiftene@info.uaic.ro

Abstract

Textual entailment recognition RTE\textsuperscript{37} (Dagan et al., 2006) is the task of deciding, given two text fragments, whether the meaning of one text is entailed (can be inferred) from the other text. This year, at our first participation in an RTE competition, we built a system targeting the scope of TE – semantic variability. The main idea of our system is to transform the hypothesis making use of extensive semantic knowledge from sources like DIRT, WordNet, Wikipedia and acronyms database. Additionally, the system applies complex grammar rules for rephrasing in English and uses the results of a module we built to acquire the extra background knowledge needed.

1 Introduction

In the textual entailment competition, participants in the evaluation exercise are provided with pairs of small text snippets (one or more sentences in English), which we term Text-Hypothesis (T-H) pairs. They must build a system that, for each pair, determines entailment or no. Complexity of this task comes from applications that need to assess such a semantic relationship between text segments: Information Retrieval (IR), Question Answering (QA), Information Extraction (IE), and Text Summarization (SUM).

This year the challenge arrived at his third edition. The precedent challenges were in 2005 and 2006. In every new competition new features have been added. In the current challenge a limited number of longer texts, i.e. up to a paragraph, was added in order to move toward more comprehensive scenarios which incorporate the need for discourse analysis.

Also the pilot task\textsuperscript{38} from this year with goal of making a three-way decision of “YES” (entails), “NO” (contradicts), and “UNKNOWN” wants to drive systems to make more precise informational distinctions. A hypothesis being unknown on the basis of a text should be distinguished from a hypothesis being shown false/contradicted by a text. Another goal of this task was to provide justifications for decisions and to explore how eventual users of tools incorporating entailment can be made to understand how decisions were reached by a system.

Section 2 briefly presents our system characteristics and results, and in Section 3 we present our conclusions and future work.

2 System presentation

The system (Iftene and Balahur, 2007) build for competition has two main parts: one for all requires pre-processing and another one (the main module) which uses the output of the first phase and obtains in the end the answers for all pairs. We can see in Figure 1 how the pre-processing is realized in parallel with MINIPAR (Lin, 1998) and LingPipe\textsuperscript{39} modules which provide the input for the core module which uses four databases: DIRT, Acronyms, Background knowledge and WordNet.

The system architecture is based on a peer-to-peer architecture (Iftene and Croitoru, 2006). It uses a common caching mechanism for large databases of Dirt and WordNet and a quota mechanism for synchronizing the ending of all processes, in order to increase the computation speed.

\textsuperscript{37} http://www.pascal-network.org/Challenges/RTE/

\textsuperscript{38} http://nlp.stanford.edu/RTE3-pilot/

\textsuperscript{39} http://www.alias-i.com/lingpipe/
The first step splits the initial file into pairs of files for text and hypothesis. All these files are then sent to the LingPipe module in order to find the Named entities. LingPipe is a suite of Java libraries for the linguistic analysis of human language. The major tools included in this suit are for sentence, parts of speech, named entities and coreference. From this suite we have used only the Named Entities recognition tool with scope to eliminate the pairs that have a named entity only in the hypothesis, concluding that there is no entailment. For instance, simple named entity recognizer for English might find the person mention John J. Smith and the location mention Washington in the text “John J. Smith lives in Washington”.

2.2 MINIPAR

In parallel, we transform with MINIPAR both the text and the hypothesis into dependency trees. MINIPAR is a broad-coverage parser for the English language. An evaluation with the SUSANNE corpus shows that MINIPAR achieves about 88% precision and 80% recall with respect to dependency relationships.

MINIPAR represents the grammar as a network, where the nodes represent grammatical categories and the links represent types of syntactic (dependency) relationships. The grammar network consists of 35 nodes and 59 links. Additional nodes and links are created dynamically to represent subcategories of verbs. The lexicon in MINIPAR is derived from WordNet.

The output produced by MINIPAR is a graph in which the nodes are the words for the parsed sentence labeled with their grammatical category and the edges are the relations between words labeled with the grammatical relationship.

In a plain text, the output for the sentence “Le Beau Serge was directed by Chabrol.” is this:

```
( E0 ( () fin C * )
1 (Le ~ U 3 lex-mod (gov Le Beau Serge))
2 (Beau ~ U 3 lex-mod (gov Le Beau Serge))
3 (Serge Le Beau Serge N 5 s (gov direct))
4 (was be be 5 be (gov direct))
5 (directed direct V E0 i (gov fin))
E2 ( () Le Beau Serge N 5 obj (gov direct) (antecedent 3))
6 (by ~ Prep 5 by-subj (gov direct))
7 (Chabrol ~ N 6 pcomp-n (gov by))
8 ( . ~ U * punc )
```

Table 1: MINIPAR output – plain text

In which we can see in the second column the words, in the fourth column the grammatical category and in the last column the grammatical relationship. The visual graph of the sentence is in the Figure 2.

For every node from the MINIPAR output (which represents a simple word belonging to a sentence), we consider a stamp called entity with three main features: the node lemma, the father lemma and the edge label (like in Figure 3).

---

40 The SUSANNE corpus is a subset of the Brown Corpus of American English
Using this stamp, we can easily distinguish between nodes of the trees, even if these have the same node lemma and the same father. In the example from Figure 2, for the “son” nodes we have two entities (Le_Beau_Serge, direct, s) and (Le_Beau_Serge, direct, obj).

2.3 The hypothesis tree transformation

Presently, the core of our approach is based on a tree edit distance algorithm applied on the dependency trees of both the text and the hypothesis (Kouylekov and Magnini, 2005). If the distance (i.e. the cost of the editing operations) among the two trees is below a certain threshold, empirically estimated on the training data, then we assign an entailment relation between the two texts.

The main goal is to map every entity in the dependency tree associated with the hypothesis (called from now on hypothesis tree) to an entity in the dependency tree associated with the text (called from now on text tree).

For every mapping we calculate a local fitness value which indicates the appropriateness between entities. Based on this local fitness an extended local fitness is built and, in the end, using all these partial values the global fitness is obtained.

For every entity from hypothesis tree which has a corresponding entity in the text tree with the same features (it can be mapped directly), we will consider the local fitness value to be 1. For entities without a direct mapping, we have the following possibilities:

- If the word is a verb in the hypothesis tree, we use the DIRT resource (Lin and Pantel, 2001) and transform the hypothesis tree into an equivalent one, with the same nodes except the verb. Our aim in performing this transformation is to find a new value for the verb which can be better mapped in the text tree.
- If the word is marked as named entity by LingPipe or if the word is a number, we try to use an acronyms’ database or obtain information related to it from the background knowledge. In the event that even after these operations we cannot map the word from the hypothesis tree to one node from the text tree, we decide the final result for this pair: No entailment.
- Else, we use WordNet (Fellbaum, 1998) to look up synonyms for this word and try to map them to nodes from the text tree.

Following this procedure, for every transformation with DIRT or WordNet, we will consider for local fitness the similarity value indicated by these resources. If after all checks we cannot map one node from the hypothesis tree, we insert some penalty in the value of the node local fitness.

- The DIRT resource

DIRT (Discovery of Inference Rules from Text) is both an algorithm and a resulting knowledge collection created by Lin and Pantel at the University of Alberta. The algorithm automatically learns paraphrase expressions from text using the Distributional Hypothesis over paths in dependency trees. A path, extracted from a parse tree, is an expression that represents a binary relationship between two nouns. In short, if two paths tend to link the same sets of words, DIRT hypothesizes that the meanings of the corresponding patterns are similar.

The DIRT knowledge collection is the output of the DIRT algorithm over a 1GB set of newspaper text (San Jose Mercury, Wall Street Journal and AP Newswire from the TREC-9 collection). It extracted 7 million paths from the parse trees (231,000 unique) from which paraphrases were generated.

For the verbs in the MINIPAR output, we extract templates with DIRT like format. For the sample output in Figure 2, where we have a single verb “direct”, we obtain the following list of “full” templates: N:s:V<direct>V:by:N and N:obj:V<direct>V:by:N. To this list we add a list of “partial” templates: N:s:V<direct>V:;V<direct>V:by:N; V<direct>V:by:N, and N:obj:V<direct>V:.

In the same way, we build a list with templates for the verbs in the text tree. With these two lists of templates we perform a search in the DIRT database and extract the “best” trimming using template type (full or partial) and the DIRT score.

http://www.acronym-guide.com

It is possible for the result of the search process to be empty. In this case, we use the corresponding noun form of the verbs and attempt to make the association between verbs and nouns.

According to the search results, we have the following situations:

a) Left – left relations similarity
   This case is described by the situation in which for the hypothesis we have the template:
   \[ \text{relation1 HypothesisVerb relation2} \]
   And for text we have the template
   \[ \text{relation1 TextVerb relation3} \]
   This is a classic case, which appears more often, in which a verb is replaced by one of its synonyms or equivalent expressions
   In this case, we transform the hypothesis tree by following two steps:
   1. Replace the relation2 with relation3,
   2. Replace the verb from the hypothesis with the corresponding verb from the text. (As it can be observed in Figure 4).

\[ \text{Figure 4: Left-left relation similarity} \]

For example, in the test set we have pair 37 with the verb “start”:

\begin{itemize}
\item \textit{T}: She was transferred again to Navy when the American Civil War began, 1861.
\item \textit{H}: The American Civil War started in 1861.
\end{itemize}

In this case, for the text we have the template \( N:s>V<\text{begin}>V:obj:N \), and for the hypothesis the template \( N:s>V<\text{start}>V:subj:N \). Using DIRT, hypothesis \( H \) is transformed into:

\[ H': \text{The American Civil War began in 1861.} \]

Under this new form, \( H \) is easier comparable to \( T \).

b) Right – right relations similarity: The same idea from previous case
c) Left – right relations similarity
   This case can be described by the situation in which for the hypothesis we have the template:
   \[ \text{relation1 HypothesisVerb relation2} \]
   And for text we have the template
   \[ \text{relation3 TextVerb relation1} \]
   In this case, we transform the hypothesis tree by following three steps:
   1. Replace the relation2 with relation3,
   2. Replace the verb from the hypothesis with the corresponding verb from the text.
   3. Rotate the subtrees accordingly: left subtree will be right subtree and vice-versa right subtree will become left-subtree (as it can be observed in Figure 5).

\[ \text{Figure 5: Left-right relation similarity} \]

For example, in the test set we have pair 161 with the verb “attack”:

\begin{itemize}
\item \textit{T}: The demonstrators, convoked by the solidarity with Latin America committee, verbally attacked Salvadoran President Alfredo Cristiani.
\item \textit{H}: President Alfredo Cristiani was attacked by demonstrators.
\end{itemize}

In this case, for the text we have the template \( N:subj:V<\text{attack}>V:obj:N \), and for the hypothesis the template \( N:obj:V<\text{attack}>V:by:N \). Using DIRT, hypothesis \( H \) is transformed into:

\[ H': \text{Demonstrators attacked President Alfredo Cristiani.} \]

Under this new form, \( H \) is easier comparable to \( T \).

d) Right – left relations similarity: The same idea from the previous case

For every node transformed with DIRT, we consider his local fitness as being the similarity value indicated by DIRT.

- Extended WordNet
   For non-verbs nodes from the hypothesis tree, if in the text tree we do not have nodes with the same
lemma, we search for their synonyms in the extended WordNet\(^ {43}\). For every synonym, we check to see if it appears in the text tree, and select the mapping with the best value according to the values from Wordnet. Subsequently, we change the word from the hypothesis tree with the word from Wordnet and also its fitness with its indicated similarity value. For example, the relation between “relative” and “niece” is accomplished with a score of 0.0786.

### Acronyms

The acronyms’ database helps our program in finding relations between the acronym and its meaning: “US - United States”, “USA - United States of America”, and “EU - European Union”. We change also the word with the corresponding expression from this database, but because the meaning is the same, we put the local fitness value on maximum i.e. 1.

### Background Knowledge

Some information cannot be deduced from the already used databases and thus we require additional means of gathering extra information of the form:

<table>
<thead>
<tr>
<th>Argentine [is] Argentina</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netherlands [is] Holland</td>
</tr>
<tr>
<td>2 [is] two</td>
</tr>
<tr>
<td>Los Angeles [in] California</td>
</tr>
<tr>
<td>Chinese [in] China</td>
</tr>
</tbody>
</table>

**Table 2: Background knowledge**

Background knowledge was built semi-automatically, for the named entities and for numbers from the hypothesis without correspondence in the text. For these named entities, we used a module to extract from Wikipedia\(^ {44}\) snippets with information related to them.

Subsequently, we use this file with snippets (Argentina in sample from Table 3) and some previously set patterns of relations between the entity in question and other named entities with the goal in this endeavor is to identify a known relation between two named entities (one of these must be our entity for which we have a problem and the other another named entity).

```
<table>
<thead>
<tr>
<th>ar</th>
<th>calling_code = 54</th>
<th>footnotes = Argentina</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>also has a territorial dispute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Argentina', , NaciĂłn Argentina (Argentine Nation) for many legal purposes), is</td>
<td></td>
</tr>
</tbody>
</table>
```

In the world. Argentina occupies a continental surface area of

<table>
<thead>
<tr>
<th>Argentina national football team</th>
</tr>
</thead>
</table>

**Table 3: Snippets extracted for Argentina**

If such a relation is found, we make the association and save it to an output file. For our case only line “Argentina [is] Argentine” is added to the background knowledge.

For the NE ‘Netherlands’, the file is larger and the result as follows:

<table>
<thead>
<tr>
<th>Netherlands [is] Dutch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netherlands [is] Nederlandse</td>
</tr>
<tr>
<td>Netherlands [is] Antillen</td>
</tr>
<tr>
<td>Netherlands [in] Europe</td>
</tr>
<tr>
<td>Netherlands [is] Holland</td>
</tr>
<tr>
<td>Antilles [in] Netherlands</td>
</tr>
</tbody>
</table>

**Table 4: Results for Netherlands**

All these relations are added to the background knowledge database and will be used at the next run. Not all relations are correct, but the relation “Netherlands [is] Holland” will help us at the next run.

Our patterns identify two kinds of relations between words:

- “is”, when the module extracts information in the form: ‘Argentina Republic’ (Spanish: ‘Republica Argentina’, IPA)’ or when explanations about the word are between brackets, or when the extracted information contains one verb used to define something, like “is”, “define”, “represent”: ‘2’ (‘two’) is a number.
- “in” when information is of the form: ‘Chinese’ refers to anything pertaining to China or in the form Los Angeles County, California, etc.

In this case, the local fitness for the node is set to the maximum value for the [is]-type relations, and it receives some penalties for the [in]-type relation.

### 2.4 Determination of entailment

After transforming the hypothesis tree, we calculate a global fitness score- using the following extended local fitness value for every node from the hypothesis - which is calculated as sum of the following values:

1. local fitness obtained after the tree transformation and node mapping,
2. parent fitness after parent mapping,

---

\(^{43}\) http://xwn.hlt.utdallas.edu/downloads.html

\(^{44}\) http://en.wikipedia.org/wiki/Main_Page
3. mapping of the node edge label from the hypothesis tree onto the text tree,
4. node position (left, right) towards its father in the hypothesis and position of the mapping nodes from the text.

After calculating this extended local fitness score, the system computes a total fitness for all the nodes in the hypothesis tree and a negation value associated to the hypothesis tree. Tests on the data have shown that out of these parameters, some are more important (1.) and some have a smaller significance (3.). Below you can observe an example of how the calculations for 3 and 4 are performed and what the negation rules are.

2.5 Edge label mapping

After the process of mapping between nodes, we check how edge labels from the hypothesis tree are mapped onto the text tree. Thus, having two adjacent nodes in the hypothesis, which are linked by an edge with a certain label, we search on the path between the nodes’ mappings in the text tree this label. (see Figure 6)

![Diagram of text tree and hypothesis tree with edge label mapping](Image)

**Figure 6: Entity mapping**

It is possible that more nodes until the label of the edge linking the nodes in the hypothesis exist, or it is possible that this label is not even found on this path. According to the distance or to the case in which the label is missing, we insert some penalties in the extended local fitness.

2.6 Node position

After mapping the nodes, one of the two following possible situations may be encountered:

- The position of the node towards its father and the position of the mapping node towards its father’s mapping are the same (left-left or right-right). In this case, the extended local fitness is incremented.
- The positions are different (left-right or right-left) and in this case a penalty is applied accordingly.

2.7 Negation rules

For every verb from the hypothesis we consider a Boolean value which indicates whether the verb has a negation or not, or, equivalently, if it is related to a verb or adverb “diminishing” its sense or not. Consequently, we check in its tree on its descending branches to see whether one or more of the following words are to be found (pure form of negation or modal verb in indicative or conditional form): “not”, “may”, “might”, “cannot”, etc. For each of these words we successively negate the initial truth value of the verb, which by default is “false”. If we have an even number of this kind of words, the final value remains “false” and in case there are an odd number of such words, the final value is changed to “true”.

Since the mapping is done for all verbs in the text and hypothesis, regardless of their original form in the text snippet, our endeavors also focused on studying the impact the original form of the verb has on its overall meaning within the text. Infinitives can be easily identified when used in the traditional form, preceded by the particle “to”. Observing this behavior, one complex rule for negation was built for the particle “to” when it precedes an infinitive form of a verb. In this case, the sense of the verb in infinitive is strongly influenced by the active verb, adverb or noun the particle “to” follows. Therefore, in the case of its being preceded by a verb like “allow, impose, galvanize” or their synonyms, or adjective like “necessary, compulsory, free” or their synonyms or noun like “attempt”, “trial” and their synonyms, the meaning of the verb in infinitive form is stressed upon and becomes “certain”. For all other cases, the particle “to” preceding a verb in infinitive form has the same behavior like the words of the previous list – it diminishes the certainty of the action expressed in the infinitive-form verb. Based on the synonyms database with the English thesaurus[^45], we built two separate lists – one of “certainty stressing (preserving)” – “positive” and one of “certainty diminishing” – “negative” words. Some examples of these words are “probably”, “likely” – from the list of

[^45]: http://thesaurus.reference.com/
“negative” words and “certainly”, “absolutely” – from the list of “positive” words.

2.8 Global fitness calculation

We calculate for every node from the hypothesis tree the value of the extended local fitness, and afterwards consider the normalized value relative to the number of nodes from the hypothesis tree. We denote this result by $TF (Total \ Fitness)$:

$$TF = \frac{\sum_{nodesH} \ ExtendedLocalFitness_{node}}{HypothesisNodesNumber}$$

After calculating this value, we compute a value $NV$ (the negation value) indicating the number of verbs with the same value of negation, using the following formula:

$$NV = \frac{Positive\_VerbsNumber}{TotalNumberOfVerbs}$$

where the $Positive\_VerbsNumber$ is the number of non-negated verbs from the hypothesis using the negation rules, and $TotalNumberOfVerbs$ is the total number of verbs from the hypothesis.

Because the maximum value for the extended fitness is 4, the complementary value of the $TF$ is $4-TF$. The formula for the global fitness used is:

$$GlobalFitness = NV * TF + (1 - NV) * (4 - TF)$$

For pair 518 from the test set:

T: The French railway company SNCF is cooperating in the project.

H: The French railway company is called SNCF.

we have extended local fitness’s like in next table:

<table>
<thead>
<tr>
<th>Initial entity</th>
<th>Node Fitness</th>
<th>Extended local fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>(the, company, det)</td>
<td>1</td>
<td>3.125</td>
</tr>
<tr>
<td>(French, company, nn)</td>
<td>1</td>
<td>3.125</td>
</tr>
<tr>
<td>(railway, company, nn)</td>
<td>1</td>
<td>3.125</td>
</tr>
<tr>
<td>(company, call, s)</td>
<td>1</td>
<td>2.5</td>
</tr>
<tr>
<td>(be, call, be)</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>(call, -, -)</td>
<td>0.0965</td>
<td>3.0482</td>
</tr>
<tr>
<td>(company, call, obj1)</td>
<td>1</td>
<td>1.125</td>
</tr>
<tr>
<td>(SNCF, call, desc)</td>
<td>1</td>
<td>2.625</td>
</tr>
</tbody>
</table>

**Table 5**: Extended local fitness calculation

$$Total\_Fitness = (3.125 + 3.125 + 3.125 + 2.5 + 4 + 3.0482 + 1.125 + 2.625)/8 = 22.6732/8 = 2.8341$$

$NV = 1/1 = 1$

$$GlobalFitness = 1 * 2.8341 + (1 - 1) * (4 - 2.8341) = 2.8341$$

Using the development data, we establish a threshold value of 2.06, and according to this, we decide that pairs above it will have the answer “yes” for entailment.

2.9 Results in RTE3

Our system has a different behavior on different existing tasks, with higher results on Question Answering (0.87) and lower results on Information Extraction (0.57). We submitted two runs for our system, with the difference residing in the parameters used in calculating the extended local fitness. However, the results are almost the same:

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>IR</th>
<th>QA</th>
<th>SUM</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run01</td>
<td>0.57</td>
<td>0.69</td>
<td>0.87</td>
<td>0.635</td>
<td>0.6913</td>
</tr>
<tr>
<td>Run02</td>
<td>0.57</td>
<td>0.685</td>
<td>0.865</td>
<td>0.645</td>
<td>0.6913</td>
</tr>
</tbody>
</table>

**Table 6**: Test results

To be able to see each component’s relevance, the system was run in turn with each component removed. The results below show that the system part verifying the NEs is the most important.

<table>
<thead>
<tr>
<th>System Description</th>
<th>Precision</th>
<th>Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without DIRT</td>
<td>0.6876</td>
<td>0.54 %</td>
</tr>
<tr>
<td>Without WordNet</td>
<td>0.6800</td>
<td>1.63 %</td>
</tr>
<tr>
<td>Without Acronyms</td>
<td>0.6838</td>
<td>1.08 %</td>
</tr>
<tr>
<td>Without BK</td>
<td>0.6775</td>
<td>2.00 %</td>
</tr>
<tr>
<td>Without Negations</td>
<td>0.6763</td>
<td>2.17 %</td>
</tr>
<tr>
<td>Without NEs</td>
<td>0.5758</td>
<td>16.71 %</td>
</tr>
</tbody>
</table>

**Table 7**: Components relevance

2.10 Building justification for answers

For the pilot task, we built two new additional modules, and for each of them we ran the system separately.

In the first one, we used the answers from the RTE3 system and split the answers in two sets of those marked with NO: contradictions (No) and the rest (Unknown). In this case, depending on which module was involved in the evaluation, we decide also the justification.

For the second one, the final score and the modules involved in evaluation were used in order to obtain both the final answer and the justification.

The results can be seen in the tables below. The F score was computed over YES and NO cases only and uses a beta value of 1/3 to give a precision value of three times the weight of recall:
Table 8: Pilot task results

<table>
<thead>
<tr>
<th>System</th>
<th>Accuracy</th>
<th>F(b=1/3)</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>System 1</td>
<td>0.569</td>
<td>0.595</td>
<td>0.547</td>
<td>0.805</td>
</tr>
<tr>
<td>System 2</td>
<td>0.471</td>
<td>0.475</td>
<td>0.437</td>
<td>0.643</td>
</tr>
</tbody>
</table>

Table 9: Table over all submitted runs

Table 9: Table over all submitted runs

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>F(beta=1/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.365</td>
<td>0.211</td>
</tr>
<tr>
<td>median</td>
<td>0.471</td>
<td>0.475</td>
</tr>
<tr>
<td>max</td>
<td>0.731</td>
<td>0.753</td>
</tr>
</tbody>
</table>

3 Conclusions

The central and the original side of the algorithm explored acquiring new information to further add to the background knowledge mechanisms and also pursued the issue of semantic variability in a new way, by proposing a set of rules for semantic equivalence. Thus, the system’s originality resides firstly in creating a part-of and equivalence ontology using an extraction module for Wikipedia data on NEs, secondly in using a distinct database of acronyms from different domains, thirdly acquiring a set of important context influencing terms and creating a semantic equivalence set of rules based on English rephrasing concepts and last, but not least, on the technical side, using a distributed architecture for time performance enhancement.

The approach unveiled a series of issues related to the dependency of analysis to parsing tools, for example separating the verb and the preposition in the case of phrasal verbs, resulting in the change of meaning. Another issue was identifying expressions that change context nuances, denoted by “positive” or “negative” words. Although we applied rules for them, we still require analysis to determine their accurate quantification.

For the future, the first concern is to search for a method to establish more precise values for penalties, in order to obtain lower values for pairs with No entailment. Furthermore, a new method is necessary to determine more precisely the multiplication coefficients for the parameters in the extended local fitness and the global threshold.

Another goal for the future is inserting the Textual Entailment system as part of a QA system in order to find the correct answers for the questions in a more precise manner. Also, the actual speed in finding answers (around 6 seconds for 800 pairs) suggests the possibility to comprehension it in a Real Time system for Question Answering.

Acknowledgements

The author would like to thank the members of the NLP group from Iasi for their help and support at different stages of the system development.

The work on this project is partially financed by Siemens VDO Iasi and by the CEEX Rotel project number 29.

References


Discourse parsing using NLP architectures

Ionuț Cristian Pistol
Faculty of Computer Science
“Alexandru Ioan Cuza” University of Iași
ipistol@info.uaic.ro

Abstract

The paper presents the principles and some implementation details of the ALPE \(^{46}\) software system intended to automate the building of modular processing architectures designed for natural language processing. The system offers many advantages over pipeline processing systems, as it is able to identify the format of annotated corpora, then to compute and run the processing steps required to bring an input file to the required output format. A graph of annotation schemas is used to represent and identify the initial and the desired schema and to compute the processing path between the two. Processing modules and resources, attached to the edges of the graph, will then be combined to perform the required transformations. The system is used to perform discourse parsing using modules previously developed. Later, the system will evolve as an on-line environment for complex linguistic processing.

1 Introduction

A NLP system which automatically determines, given an input corpus of consistent format and a required output format, the required processing modules, resources and order of processing could provide a quick and easy to use environment for NLP. The strongest advantage of such a system is that the user intervention is minimal, limited only to the specification of the input data and output format, in contrast to a “classical” pipeline processing system in which the user must manually specify and run, step by step, the modules and resources to be used.

Cristea et al. (2006) describe a methodology which allows automatic pipeline type configuration of processing sequences using a lattice of XML annotation standards. In (Cristea and Butnariu, 2004), different layers of annotation over a corpus are codified as a hierarchy of annotation standards, (directed acyclic graph), in which nodes are annotation schemas and edges are subsuming relations between schemas. In this hierarchy, a lower node indicates that the corresponding annotation standard contains more information then a higher situated node. Depending on the annotation schemas that fill up the hierarchy, the model allows classification, simplification and merging operations.

Automatic translation, summarization, generation and understanding discourse does benefit significantly from having available the structural representation of the interpreted text. In (Cristea et al., 2005) is described a discourse parsing system used to obtain summaries focused on specific characters of a free text. Summaries are extracted from discourse structures which differ from RST structures by the fact that the trees are binary and lack relation names. The discourse tree structures are obtained by combining constraints given by cue phrases with constraints coming from the exploitation of cohesion and coherence properties of the discourse (as proved by Veins Theory).

Section 2 of the paper points the challenges faced in developing complex NLP systems, section 3 of this paper describes ALPE’s hierarchical model of annotation schemas and some details concerning the modules and resources, as well as the processing path. Section 4 briefly presents the discourse parsing system and its processing modules. Section 5 presents the current development stage of the ALPE system, the evaluated feasibility of the approach in practical settings and the planned integration of the parsing system in ALPE, as well as the further planned developments.
2 Annotation systems

One of the latest developments in Computational Linguistics, and one which promises to have a significant impact for future linguistic processing systems, is the emerging of linguistic annotation meta-systems, that make use existing processing tools and implement some sort of processing path, pipelined or otherwise.

The increasing size of modern corpora poses some new challenges to the software that is used to process them. Not only is the sheer amount of data becoming a problem, but also the handling of additional administrative information available for texts. While there has been a lot of work on standardisation of data formats, no such efforts have been undertaken on the software side. Each system using annotated corpora basically has to develop their own processing tools, as previous programs become unable to rise to the new challenges or comply to the new input and output formats. In order to avoid duplication of efforts, and to find a solution that is acceptable to everybody involved, it is necessary to define some standardized interfaces which programs can implement. Just as there has been agreement on what aspects of a corpus need to be encoded, and how they should be encoded, a kind of ‘Text Processing Initiative’ would need to define a basic set of actions on texts.

Different tools would implement different aspects of this set of actions; and also with different choices regarding trade-off decisions such as speed vs. space. By combining several tools one could end up with the ideal customized corpus processing tool, geared to specific preferences relevant for a certain kind of research.

The Corpus Universal Examiner (CUE) (Mason, 1998) was the first implementation of such a system. It planned to provide a programming environment for the development of applications specialized in processing linguistic corpora, using facilities implemented in CUE, such as optimizations for working with large files (such as those in a corpus) or pattern-matching. CUE was not fully developed, as further projects took over the main ideas, but tried to provide means to combine existing processing tools, rather than creating new ones in a dedicated programming environment.

The ATLAS framework (Bird et al., 2000) also provided an architecture targeted at facilitating the development of linguistic annotation applications, but also focused on creating a logical layer for annotations, in which different formats would be converted and combined according to a required schema.

CUE and ATLAS were also two of the starting points of GATE, a specialized annotation system developed at the Sheffield’s University Natural Language Processing Research Group. GATE was developed around the theory of Software Architecture for Language Engineering (SALE) (Cunningham, 2000), which formalizes the basic requirements for a processing architecture for linguistic corpora.

GATE as an architecture suggests that the elements of software systems that process natural language can usefully be broken down into various types of component, known as resources4. Components are reusable software chunks with well-defined interfaces, and are a popular architectural form, used in Sun’s Java Beans and Microsoft’s .Net, for example. GATE components are specialized types of Java Bean, and come in three flavors:

- LanguageResources (LRs) represent entities such as lexicons, corpora or ontologies;
- ProcessingResources (PRs) represent entities that are primarily algorithmic, such as parsers, generators or ngram modellers;
- VisualResources (VRs) represent visualisation and editing components that participate in GUIs.

Collectively, the set of resources integrated with GATE is known as CREOLE: a Collection of REusable Objects for Language Engineering. All the resources are packaged as Java Archive (or ‘JAR’) files, plus some XML configuration data. The JAR and XML files are made available to GATE by putting them on a web server, or simply placing them in the local file space.

When using GATE to develop language processing functionality for an application, the developer uses the development environment and the framework to construct resources of the three types. This may involve programming, or the development of Language Resources such as grammars that are used by existing Processing Resources, or a mixture of both. The development environment is used for visualisation of the data structures produced and consumed during processing, and for debugging, performance measurement and so on.
Another development projects in this direction is IMB’s Unstructured Information Management Architecture (UIMA) (Ferrucci and Lally, 2004) a research project started in 1999, which has in 2004 provided a software platform for combining semantic analysis and search components, such as indexing tools. It developed into a full Java implementation, providing classes and methods for several processing paths and tools available.

One of the latest automated processing system is LiveTree (Thione et al., 2004). LiveTree is an integrated workbench for supervised and unsupervised creation, storage and manipulation of the discourse structure of text documents under the ULDM (Polanyi et al., 2004), a discourse parsing system. The LiveTree environment provides tools for manual and automatic U-LDM segmentation and discourse parsing. Like RSTTool (O’Donnell, 1997), LiveTree provides support for segmentation, marking structural relations among segments, and creating and editing discourse relations. Similar to the DLTAG system described in (Forbes et al 2003) LiveTree is an experimental discourse parser implementing a theory of sentential and discourse relations. However, LiveTree is also a complete document handling and manual and automatic discourse parsing system. Various applications are supported as web services. Accordingly, LiveTree serves as both the user interface and theory development environment for PALSUMM, a text summarization system.

Many processing architectures using several linguistic annotation modules have been recently developed and used. For instance, DeepThought (Callmeier et al., 2004) is a framework for integrating shallow with deep linguistic processing modules, by using linguistic annotation in various formats. The framework is implemented in the HeartOfGold (Schäfer, 2006) system, developed as a front-end for a HPSG parser. This system uses elaborate schema descriptions to allow the automatic extraction of linguistic information from the output of processing modules. Annotation Graphs is a formal framework for representing linguistic annotations of time series data. Annotation graphs abstract away from file formats, coding schemas and user interfaces, providing also to a user a logical layer for annotation systems. GATE (Cunningham et al., 2002, 2003), is a versatile environment for building and deploying NLP software and resources, allowing for the integration of a large amount of built-ins in new processing pipelines that can process single documents or corpora. The user can configure new architectures by selecting the desired modules from a repository, as parts of a processing chain.

All these approaches provide annotation models and solutions for manually configuring processing chains over text files which observe XML mark-up. However, to our knowledge, there is no serious account on automatizing the designing of linguistic processing in language technology. The need for such a perspective is made more and more evident by the new challenges of processing large amounts of linguistic data with the intent of accessing the semantic content, and of making it available to applications of different nature, as imposed by Semantic Web. Our approach departs from the traditional setting of manually assembling components in chains of processors aimed to fulfill certain tasks, towards the fulfillment of the demand to process linguistic data of diverse nature quickly and efficiently. The new perspective is one in which Web services are used to enhance the use and access to language resources in all forms and to account for processors which are not available locally. Although merging of annotation, including that of different phenomena, and simplification of annotation has been performed by other systems as well, our model makes them elementary steps in a powerful and automated processing flow.

3 ALPE

In (Cristea and Butnariu, 2004), different layers of annotation over a corpus are represented as a hierarchy of annotation schemas. Each node in the hierarchy contains a list of XML tags, each characterized by a name, a list of attribute names, and possible restrictions denoting identity of some of its attributes values with values of attributes of other tags in the hierarchy. The parenthood relationship places the schemas described in this way in a hierarchy, which is a directed acyclic graph whose node names are unique symbols. If a node $A$ is directly linked to a node $B$, then it is said that $A$ subsumes $B$ in the hierarchy (therefore $B$ is a descendant of $A$). This happens if and only if:

- any tag-name of $A$ is also in $B$;
- any attribute in the list of attributes of a tag-name in $A$ is also in the list of attributes of the same tag-name of $B$;
any restriction relation which holds in A also holds in B.

A consequence of the subsuming relation is the inheritance of the features (seen here as tags and their attributes) from the parents. Therefore, a hierarchical relation between a node $A$ and a descendant $B$ describes $B$ as an annotation standard which is more informative than $A$ and/or defines more semantic constraints. The subsuming relation is transitive, reflexive and asymmetrical.

In Figure 1, an example of a hierarchy of annotation schemas is represented. The root ($SCH$-Root) represents the text with no annotation. The next level contains three schemas, $SCH$-TOK, $SCH$-SEG and $SCH$-PAR. $SCH$-TOK represents a level of tokens annotation, $SCH$-SEG marks borders between elementary discourses. $SCH$-POS, placed under $SCH$-TOK, adds information related to the part-of-speeches. The $SCH$-POS has as descendent the $SCH$-NP (for the noun phrases level) and $SCH$-VP (for verb phrase). $SCH$-SEG-NP-VP is a schema marking simultaneously noun phrases, verb phrases and discourse units boundaries. It adds no new markings to those inherited from its three parents. This hierarchy is a simplified version of that which will be implemented in ALPE for the discourse parsing system (see sections 3 and 4).

Figure 1 shows an example of an ALPE type hierarchy. The hierarchical links from parent nodes to descendants are marked as oriented edges (arrows) and represent simultaneously subsuming relations and processing steps. The labels indicate the names of the modules used to add annotation information to the texts. The names of processes are marked on the edges, while the symbol $\emptyset$ marks the empty process (no contribution of new tags/attributes).

In the ALPE system, this hierarchy of schemas is augmented with processing power. The input file can contain any annotation, as this file will be automatically classified in an annotation schema and integrated in the graph hierarchy. Then, if there are requirements of a specified output schema, the augmented graph can compute the processing sequences needed to transform the input file onto the output file. If the modules associated to the edges are present, then the actual processing can be started. There are also improvements concerning the pipeline construction: in GATE (Cunningham et al., 2002, 2003), the user must build the pipeline (select the resources needed to run the application, ensure that the components selected are listed in the correct order for processing, ensure that any parameters necessary are set for each processing resource), while in our approach a processing flow is identified automatically.

The tree operations used in the processing path on the graph are simplification, pipelining and merging. The input and the output file are associated with a start, respectively a destination node, both belonging to the graph. The input file contains the restriction of the start node. By applying the processing paths to the input file, the output file, respecting the restrictions of the destination node, is eventually produced.

Having a process path applied to a node, the output is a flow. Seen on the graph, a flow can be associated with the head of the arrow pointing to a node, if we imagine that through each processing edge the information actually “flows”. Flows must be seen as summing-up sequences of processing steps. When applied to an input file, a flow transforms that file by adding or deleting some markups. We will denote by $f(A)$ the flow produced when the input node is $A$. All three operations which we will define below produce information flows. Trivially, a null flow leaves an input file unmodified. We will denote the null flow with $f^\emptyset$.

If $n$ is a node, we will write $f^\emptyset(n)=n$, meaning that the null flow leaves an input file observing the restrictions imposed by the scheme $n$ unmodified. If $A$ and $B$ are the start and destination nodes of a
processing path, then there should be exactly a flow $f$ such that $B = f(A)$.

The first operation is simplification. The node $A$ is simplified in the sense $B$, if $B$ subsumes $A$, hence there is a directed path in the subsumption hierarchy from $B$ to $A$. By simplifying a file $A$ in the sense $B$, all annotations not contained in the node $B$ are deleted.

The pipeline operation appears if in the augmented graph exists an edge linking $A$ to $B$, marked with a process $p$. We say then that $A$ pipelines to $B$ by $p$. Pipelining an annotation standard $A$ to a destination standard $B$ transforms the schema $A$ using the module $p$ according to the restrictions of $B$. The "pipeline" operation is associative, non-commutative and it has the empty process $\emptyset$ as identity element.

The merge operation (a commutative and associative operation, having the null flow $\emptyset$ as its identity element) can be defined in nodes pointed by more than one edge on the hierarchical graph. Further details and examples concerning flow computation algorithm can be found in (Cristea et al., 2006).

After the flow is computed, the next phase is the execution of the resulting flow. This consists in the execution of a series of modules, each receiving its input from previous ones or from the input corpora. There are two types of modules used in ALPE:

- **ALPE core modules**: which perform merging, simplification and schema identification operations, as well as language identification and tokenization. These modules will be included in all ALPE adaptations, together with modules specific to the respective system. These modules are language independent and do not require any additional resources except language models for language identification.

- **System-specific modules**: modules implemented for various linguistic processing tasks, either independent or as part of a larger system. As part of integrating them in ALPE, some modules will require minor modifications in order to conform to a specified general format, basically they have to be command line executable and they have to have clearly specified input and output formats. Most modules also require specialized resources, like language models for the POS-tagger, lists of discourse markers for the segmenter, models of anaphora resolution for the AR module, etc.

If a processing module requires that some language-specific parameters are specified, such as a language model or a specific gazetteer, these can be identified automatically using an available language identification tool with 98% accuracy.

The execution of the flow is basically the running of a series of modules, each receiving an input file and producing the input file for the next module appearing in the flow. The compatibility of the different inputs and outputs is assured by the computation of the flow and by the design of the individual processing modules.

The information regarding the modules and their corresponding edges in the graph is contained in another XML file, codifying a series of tags `<MOD>`. An example of one such tag is:

```
<MOD name="tokenizer" start="SCH-Root" dest="SCH-TOK" app="../res/tokenizer.exe"/
lang="en,ro">
```

The 'start' and the 'dest' attributes identify the starting node and the destination node of the corresponding edge and the 'app' attribute gives the location of the respective module. When a certain module is required to process an input file, a batch file is generated and executed. This batch file provides to the module information regarding the location and the language of the input file as parameters of its execution.

We considered adopting the CREOLE description format used in GATE, but it proved inadequate for our requirements. The main problem is that CREOLE was designed to describe a more general type of resources, concerning corpora, software and application data. For our system, we will provide a semi-automated process of describing modules and additional resources, similar to the one used to identify input files. For instance, if a new processing module is connected to an edge on the graph, the format of sample input and output files is identified automatically and comparing these some assumptions can be made regarding the transformations made.
4 The discourse parsing system

In (Cristea et al., 2006) is described the architecture of a system that combines a pipeline style of processing the text with a parallel and an incremental one, with the aim to obtain an RST-like discourse structure that marks the topology and nuclearity, while ignoring the names of the rhetorical relations. Such trees are then used to compute focused summaries on searched discourse entities. In the process of building discourse trees are considered properties of the relationship between reference chains and the discourse structure as well as between global discourse structure and the smoothness of centering transitions. Both reference chains and centering transitions are related with veins expressions computed following the veins theory (VT) (Cristea et al., 1998).

First, the text is POS tagged, and then a syntactic parser (FDG) is run over it. Further, the process is split into two flows: one that segments the sentences into elementary discourse units (edus) and then constructs elementary discourse trees (edts) of each sentence, and another that detects NPs and then runs an anaphora resolution engine to detect coreferential relations. Intermediate files in the processing flow are in the XML format. When two processes join, the resulted files are merged into a single representation. An edt is a discourse tree whose leaf nodes are the edus of one sentence. Sentence internal cue-words/phrases trigger the constituency of syntactically edts from each sentence (Marcu, 2000). For each sentence in the original text a set of edts is obtained. At this point a process that simulates the human capability of incremental discourse processing is started. At any moment in the developing process, say after n steps corresponding to the first n sentences, a forest of trees is kept, representing the most promising structures built by combining in all possible ways all edts of all n sentences. Each such tree corresponds to one possible interpretation of the text processed so far. Then, at step n+1 of the incremental discourse parsing, the following operations are undertaken: first, all edts corresponding to the next sentence are integrated in all possible ways onto all the trees of the existing forest; then the resulted trees are scored according to four independent criteria, sorted and filtered so that only a fraction of them is retained (again the most promising after n+1 steps). From the final wave of trees, obtained after the last step, the highly scored is considered to be the discourse structure. Summaries are then computed on this tree.

The process to obtain discourse trees using this system is language independent in itself; however the individual modules are language-dependant. At the moment only English modules are available, with modules for the Romanian language being almost completed as well. If the required annotation modules are available, then the same procedure can be run on additional languages. This system is ideal as a test bed for automated NLP architectures, as it combines complex processing with multilinguality and modularity.

5 Conclusions

In its current state, the described system is in a fairly advanced implementation phase. The hierarchy is functional and able to classify input files in an existing node or to create a new node to fit the new annotation schema. The Simplify and Merge modules are also available, as are most of the processing modules visible on the hierarchy in Figure 1.

There are some very reliable open-source modules of GATE which we intend to make available in our system. Although each individual module can be improved, or a completely new method implemented, our system counts on improving Natural Language Processing methods by creating a dynamic and reliable environment for combining processing tools and annotations, not by implementing a new set of tools. After a complete implementation is available, it will provide means to integrate any new module made available, and it could even allow each user to choose from different modules that perform the same task.

The involvement in various research projects required that the main focus of the development done so far was in creating and using project-specific processing flows. These allowed testing of various processing modules and the development of various wrappers. The future focus in the development will be the creation of user-friendly interfaces and of modules that will allow user-specified tasks and the automated creation of new hierarchies. An initial version, using as base-hierarchy the parsing system described in section 4, will be available by the end of the year (at the latest). As evaluation, we will compare ALPE processing using available
modules with using these modules independent of ALPE. ALPE should significantly reduce the additional work necessary in order to reach the desired result.

In a further development, an on-line version will be made publicly available. This version will eventually develop into a meta-system for linguistic processing, allowing users to add new formats for linguistic corpora and new processing tools, which will be automatically added either to a global ALPE hierarchy or to a user-specific hierarchy. The global ALPE hierarchy can be viewed as a universal linguistic processor, using any available tools to produce any annotation required.

References


Ferrucci D. and Lally A. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. In Natural Language Engineering 10, No. 3-4, 327-348


Pronominal Anaphora Generation in Human-Computer Dialogue, Using a Semantic-Pragmatic Approach

Vladimir Popescu\textsuperscript{1,2}
\textsuperscript{1}Laboratoire d’Informatique de Grenoble, Grenoble Institute of Technology, France
\textsuperscript{2}Faculty of Electronics, Telecommunications and Information Technology, University “Politehnica” of Bucharest, Romania
vladimir.popescu@imag.fr

Abstract

In currently available human-computer dialogue systems, the answer generation component is usually realized in ad-hoc manners, via template-based approaches, which are rather inflexible and unable to take into account the context of the interaction. However, in order to achieve a better relevance of the answers produced by the machine, with regard to the users that it interacts with, a more “sophisticated” natural language generation (NLG) component is necessary. This is why a language generator that integrates several processing stages in generation becomes interesting in human-computer dialogue applications. In order for such a language generator to be built, a generic (i.e., application-independent) architecture has been designed; in this framework, a component responsible with pronominal anaphora generation is described in this paper. Although this issue has been thoroughly mitigated, consideration of pragmatic aspects, such as discourse structure or semantic coherence related to the context of the dialogue, has been paid less attention than purely linguistic aspects in this regard. Thus, the novelty of the approach presented in this paper is twofold: (i) the anaphora generator relies on pragmatic constraints quantified in the rhetorical structure, and (ii) the generator is independent of the actual formalism used for surface realization, the usage of several pure textual generators being thus envisageable. The paper provides the formal details of the pronominal anaphora generator, along with an extended example illustrating the approach.

1 Introduction

The goal of this paper is to describe the processing steps involved in pronominal anaphora generation for human-computer dialogue, motivated by pragmatic aspects related to user interaction context. The idea of generating pronominal anaphora via pragmatically-motivated approaches is not new; for current research in this respect we can refer to (Striegnitz, 2005) or to (Danlos and El-Ghali, 2002).

The novelty of the approach described in this paper resides in that only pragmatic (viz. discourse structure priorly computed) and semantic (entities in language independent discourse and task ontologies) knowledge is used, in an application independent manner. This approach continues previous research of our team, in order to obtain a natural language generation module, integrated to a spoken human-computer dialogue system (Caelen and Xuereb, 2007). Thus, particular features of dialogue are taken into account, such as specific rhetorical structure entailed in this context, between speech turns came from different speakers (dialogue partners).

The genericity of the approach used in pronominal anaphora generation has some limits, though; in order to handle anaphora generation in utterances, one has to dwell into language-specific details, since, although a rather pragmatically-motivated phenomenon, anaphora are constrained by language particularities. Thus, we try to
combine a set of rules handling syntactic patterns in French language via a deterministic grammar, with discourse specifications came from a rhetorical structuring component (Popescu et al., 2007).

In order to accomplish this, we consider that the head element of an utterance is the verb, which instantiates a predicate in the domain ontology. For the verbs we take into account a set of classes, defined in this paper. The point is that these classes induce transformations on the rules in the grammar, stemming to yet more constraining rules (compared to those initially in the grammar). Furthermore, in regard with the degree of pronominal anaphora usage, syntactic patterns change, in a manner that will be presented in detail.

In fact, purely linguistic (viz. morphological, syntactic) aspects concerning anaphora generation are not in focus for us, since the issue of anaphora generation has been thoroughly mitigated from a linguistic point of view, either for generation (Reiter and Dale, 2000), (Striegnitz, 2005), or for interpretation (Asher and Lascarides, 2003). Instead, pragmatic aspects related to discourse structure in dialogue, or to the role and social status of the dialogue partners are of main importance to our research.

More specifically, we consider constraints induced by the rhetorical structuring of dialogues (formalized in terms of Segmented Discourse Representation Theory - SDRT) and by the social relationship and level of expertise (in regard with the topic of the dialogue) characterizing the dialogue partners. These pragmatic constraints operate a certain filtering on pronominal anaphora generation, which can, for instance, be blocked by certain dialogue contexts, as we will see in the paper.

The paper is structured as follows: the next section presents the pronominal anaphora generation rules in detail, specifying the generation grammar, along with fundamental syntactic patterns for non-interrogative and for interrogative utterances in French language and the manner in which these patterns are constrained by the class of the verb instantiating the predicate in each utterance; the third section presents in detail the pronominal anaphora generation algorithm, showing how the pronominal anaphora generation rules are triggered by constraints induced by the discourse structure; the fourth section shows an extended example illustrating the pronominal anaphora generation mechanism; the last section concludes the paper and provides pointers to further work.

2 Pronominal Anaphora Generation Rules

2.1 Generation Grammar

In this section a set of deterministic rules connecting entities (semantic categories) in the utterances to actual words in natural language is given; the entities in the utterances are in fact semes defined in the task ontology (Popescu et al., 2007).

The grammar rules are specified using standard EBNF (“Extended Bakhsh-Naur Form”) notation, with the following particularities:
- non-terminals are delimited by squared brackets (“[...]”);
- terminals represented by words in natural language are represented as such, without any delimiters;
- non-terminals of the type $A_{\text{entity}}$ represent anaphora for entities of the type entity, where entity is one of the following semantic categories: $[\text{agt}]$, $[\text{obj}]$, $[\text{pat}]$ or $[\text{mod}]$. These semantic categories are defined below:
  - $(\text{pred})$ - the predicate of the utterance,
  - $[\text{agt}]$ - the agent, animated entity performing the action specified by the predicate of the utterance,
  - $[\text{obj}]$ - the object, inanimate entity undergoing the action performed by the agent in the utterance,
  - $[\text{pat}]$ - the patient, animated or inanimate entity enduring the effects of the action performed by the agent on the object(s) in the utterance,
  - $[\text{mod}]$ - the modifier, specifier of the action indicated by the predicate of the utterance; it is instantiated as adverb.

Some rules in the pronominal anaphora generation grammar, in French language, are given below:

1. $[\text{agt}] \rightarrow [\text{agt}] | ([\text{det}], [\text{agt}]) | [A_{\text{agt}}];$
2. $[A_{\text{agt}}] \rightarrow \text{je} | \text{tu} | \text{il} | \text{elle} | \text{nous} | \text{vous} | \text{ils} | \text{elles} | \text{celui-ci};$
3. $[\text{obj}] \rightarrow [\text{obj}] | ([\text{det}], [\text{obj}]) | [A_{\text{obj}}];$
4. $[A_{\text{obj}}] \rightarrow \text{le} | \text{la} | \text{les} | \text{en};$
5. $[\text{pat}] \rightarrow [\text{pat}] | ([\text{det}], [\text{pat}]) | [A_{\text{pat}}];$
6. $[A_{\text{pat}}] \rightarrow \text{me} | \text{te} | \text{lui} | \text{nous} | \text{vous} | \text{leur}.$

Further on we will observe the role of this grammar in controlling pronominal anaphora generation.
2.2 Syntactic Pattern - Non-interrogative Utterances

Given the fact that the dialogues concerned by our research are between human and computer, task dependent, globally coherent (rhetorically structured) and locally relevant (expressive), in the sense that Grice’s maxims are respected (Reiter and Dale, 2000), we assume that, in French language, the generator can produce utterances according to the following syntactic pattern:

\[
\text{[agt]} \langle \text{pred} \rangle \text{[obj]} \text{[pat]} \text{[mod]}.\]

Examples of utterances, in French \(^{47}\), satisfying to this pattern, are given below (several semantic entities may be void, i.e., absent in the actual linguistic realization):

Pierre part à Paris demain.
(Pierre goes to Paris tomorrow.)

Pierre parle à Marie doucement.
(Pierre speaks to Marie gently.)

From these examples we observe that the syntactic pattern does not cover all the possible expression situations, for instance in the last utterance above, in French one would say rather “Pierre parle doucement à Marie” (in English, Pierre speaks gently to Marie), involving an inversion between the patient and the modifier, when the object is missing. This is why addition of several extra syntactic patterns, driven by activation rules, may be appropriate in the future.

According to whether one or all the entities of the type agent, object, patient and modifier are realized through pronominal anaphoric constructions, the syntactic pattern changes, following a set of rules, presented in this section. In order to formalize these rules, we denote by “∀”, all possible values, authorized by the rules in the pronominal anaphora generation grammar.

One such rule, governing the transformations on the syntactic pattern, for non-interrogative utterances is given below, under the form \(^{48}\):

\[
equals([\text{agt}], \forall) \land \equals([\text{obj}], [A_{\text{obj}}]) \land \neg\equals([\text{pat}], [A_{\text{pat}}]) \land \neg\equals([\text{mod}], [A_{\text{mod}}]) \Rightarrow [\text{agt}] [\text{obj}] [\text{pred}] [\text{pat}] [\text{mod}].
\]

These rules show how the decision to generate in anaphoric manner entities in utterances induces transformations on the syntactic pattern.

2.3 Syntactic Patterns - Interrogative Utterances

We suppose that interrogative utterance aim, from the point the vue of the machine, at specifying certain elements in an utterance came from the user, in order for this utterance to become semantically complete (Caelen and Xuereb, 2007). This stems to the initialisation of the variables defined and non-initialised in utterances produced by the user. These variables can concern either the agent, or the patient, the object, and the modifier. We assume that the predicate may not be replaced by an anaphoric construction in interrogative utterances (by virtue of Grice’s maxims (Asher and Lascarides, 2003), logic formulae specifying the communicative intention to be put in surface form have at least the predicate present, initialised) and that several of the entities agent, patient and modifier may be replaced by anaphoric constructions.

For the ease of comprehension, we consider first a non-interrogative, assertive utterance completely specified, i.e., having all the semantic entities initialised:

Pierre donne le livre à Jeanne demain.
(Pierre gives the book to Jeanne tomorrow.)

For this utterance several types of interrogative utterances may exist, according to the set of entities non-initialised in preceding non-interrogative utterances; hence:

1. if the agent is not initialised, the interrogative utterance trying to determine the initialisation of the variable specifying the agent might be:

   \[\text{Qui donne le livre à Jeanne demain?}\]
   (Who gives the book to Jeanne tomorrow?)

2. if the object is not initialised, the interrogative utterance trying to determine the initialisation of the variable specifying the object might be one of the following:

   \[\text{Pierre donne quoi à Jeanne demain?}\]
   (Pierre gives what to Jeanne tomorrow?)

3. if the patient is not initialised, the interrogative utterance trying to determine the initialisation of the variable specifying the patient might be:

   \[\text{À qui Pierre donne le livre demain?}\]
   (To whom does Pierre give the book tomorrow?)

\(^{47}\) English translations of these utterances are provided, in italics, below the French versions.

\(^{48}\) The predicate \(\text{equals()}\) in the discourse ontology (Popescu \textit{et al.}, 2007) specifies that its left-hand argument is equal to its right-hand argument.
4. if the modifier is not initialised, the interrogative utterance trying to determine the initialisation of the variable specifying the modifier might be a temporal, spatial or modal adverb, inducing several syntactic patterns:

- if the unknown modifier regards the time of the action, an appropriate utterance might be:

  \[ \text{Quand est-ce que Pierre donne le livre à Jeanne ?} \]

  (When does Pierre give the book to Jeanne?)

  This kind of utterance is obtained applying the following rule:

  \[
  \text{equals}([\text{mod},"?"]) \land \text{equals}(\Delta(\alpha), "?") \Rightarrow ([\text{mod}] \rightarrow [\text{A}_{\text{mod}}]) \land ([\text{A}_{\text{mod}}] \rightarrow \text{quand}) \land ([\text{A}_{\text{mod}}] \{\text{est-ce que}\} [\text{agt}] [\text{pred}] [\text{obj}] [\text{pat}]);
  \]

- if the modifier concerns the place of the action, the anaphor will be, instead of “quand” (when), “où” (where), the anaphor generation rule remaining the same as above; nevertheless, as it will be seen further, the class of the verb instantiating the predicate of the utterance influences the new syntactic pattern;

- if the unknown modifier concerns the manner in which the action is performed, the anaphor will be “comment” (how) instead of “quand”.

  When several entities are not initialised in a non-interrogative utterance preceding an interrogation, we assume that the machine, through each question, tries to initialise one variable at a time. This is justified by the fact that the machine asks questions only in order to complete either the (incomplete) plans for finalizing tasks. In both cases, this eventually boils to initialisations of the logical variables involved.

2.4 Verb Classes - Effects on Pronominal Anaphora Generation Rules

To each predicate in an utterance (also corresponding to a predicate in the task ontology (Popescu et al., 2007), (Caelen and Xuereb, 2007) a verb corresponds; the verbs are clustered in eight classes. This taxonomy is made with respect to the presence or the absence of the semantic entities agent, object and patient (introduced above). Thus, the eight verb classes are as follows:

(i) VTD - direct transitive verbs (the agent, the object and the patient exist); example: donner (to give);

(ii) VTI 1 - indirect transitive verbs 1 (the patient does not exist, whereas the agent and the object exist); example: manger (to eat);

(iii) VTI 2 - indirect transitive verbs 2 (the object does not exist, whereas the agent and the patient exist); example: maquiller (to make up);

(iv) VIT - intransitive verbs (the object and the patient do not exist, whereas the agent exists); example: dormir (to sleep);

(v) VI 1 - impersonal verbs 1 (the agent, the object and the patient do not exist); example: pleuvoir (to rain);

(vi) VI 2 - impersonal verbs 2 (the agent and the object do not exist, whereas the object exists); example: faire beau (to be a fine time);

(vii) VI 3 - impersonal verbs 3 (the agent does not exist, whereas the object and the patient exist); example: falloir (to need);

(viii) VI 4 - impersonal verbs 4 (the agent and the object do not exist, whereas the patient exists); no example exists in French or English, but for instance in Romanian some multiword expressions featuring such a verb exist, e.g. a [\text{A}_{\text{pat}}] ploua în gât.

The class of the verb materializing the predicate in an utterance constrains the set of values that the building entities in the syntactic pattern ([\text{agt}], [\text{obj}], [\text{patient}] and [\text{mod}]) may take. Thus, using the same notational conventions as above, we can state, in a compact manner, the relationship between the class of the verb and the values of the entities in the syntactic pattern. This relationship is specified by a set of rules such as:

\[
\text{equals}([\text{pred}), \text{VTD}] \Rightarrow \neg \text{equals}([\text{agt}], \emptyset) \land \neg \text{equals}([\text{obj}], \emptyset) \land \neg \text{equals}([\text{pat}], \emptyset) \land \text{equals}([\text{mod}], \forall).
\]

We can see that, in fact, the class of the verb simplifies the syntactic pattern to be generated in anaphoric manner. However, for interrogative utterances, the class of the verb may entail modification in the syntactic pattern; thus, for intransitive verbs, a question, targeted towards the initialisation of a modifier, can be realized as specified by the following rule:

\[
\text{equals}([\text{mod},"?"]) \land \text{equals}([\text{pred}),\text{VIT}] \Rightarrow ([\text{mod}] \rightarrow [\text{A}_{\text{mod}}]) \land ([\text{A}_{\text{mod}}] \rightarrow \text{quand} | \text{où} | \text{comment}) \land ([\text{agt}] [\text{pred}] [\text{A}_{\text{mod}}]).
\]
3 Pronominal Anaphora Generation Algorithm

3.1 Pragmatic Constraints

We assume that the starting point for pronominal anaphora generation is a segmented discourse structure (SDRS), came from a rhetorical structuring component (Popescu et al., 2007). This discourse structure is computed in the framework of Segmented Discourse Representation Theory (SDRT), simulated in a first-order logic formalism; it is stated by rhetorical relations, connecting the (labeled) utterances in dialogue.

In order for the rhetorical structure (named Segmented Discourse Representation Structures - SDRS) to be taken into account in anaphora generation, a grammar for representing it (using information provided by the rhetorical structuring component) is specified. Such a grammar assumes that available information in an SDRS came from the rhetorical structuring component concerns: (i) the number of utterances in the discourse structure, denoted $SDRS$, (ii) the number $R$ of rhetorical relations connecting the utterances, (iii) the set $\Pi$ of labels $\pi$ of the utterances in the discourse structure, (iv) the set $\Pi$ of labels of the rhetorical relations$^{49}$, (v) the logic forms $\Pi$ of the utterances in the discourse structure, (vi) the semantics (expressed in a first-order logic formalism (Popescu et al.2007)) $\Sigma$ of the rhetorical relations connecting the utterances in the discourse structure.

Thus, according to the discussion provided above, the SDRS representation grammar is specified by the following rules:

1. $SDRS \rightarrow (D, R, \Pi, P)$;
2. $\Pi \rightarrow \{\pi_1, ..., \pi_p\}$;
3. $\pi_i : i \in \{1, ..., D\} \rightarrow K_j : i \in \{1, ..., D\}$;
4. $P \rightarrow \{p_1, ..., p_p\}$;
5. $p_j : j \in \{1, ..., R\} \rightarrow \rho_j (\pi_i, \pi_i) : j \in \{1, ..., R\}; i, k \in \{1, ..., D\}$;
6. $p_j : j \in \{1, ..., R\} \rightarrow \Sigma_j : j \in \{1, ..., R\}$.

Concerning the set of possible rhetorical relations connecting pairs of utterances, a set of 17 rhetorical relations have been considered (Popescu et al., 2007), out of the 35 proposed in vanilla SDRT (Asher and Lascarides, 2003). These rhetorical relations are: (i) **dialogic** relations - Q-Elab (Question Elaboration), IQAP (Indirect Question-Answer Pair), P-Corr (Plan Correction), P-Elab (Plan Elaboration), Background_q (Background Question), Elab_q (Elaboration Question), Narration_q (Narration Question), QAP (Question-Answer Pair), ACK (Acknowledgement) and NEI (Not Enough Information), and (ii) **monologic** relations - Alternation, Background, Consequence, Elaboration, Narration, Contrast and Parallel. All these rhetorical relations are simulated in a first-order logic formalism, so that they can be computed in an efficient manner for task-oriented dialogues (Popescu et al., 2007); however, the simulation preserves the informal semantics of each rhetorical relation, as specified in vanilla SDRT (Asher and Lascarides, 2003).

As we will see in the next subsection, the discourse structure imposes constraints on anaphora production, since a semantic entity in an utterance due to be generated (cf. § 2, passim) can be realized in anaphoric manner only if the following two conditions are satisfied:

- an explicitly realized referent is already introduced in a previous utterance in the current SDRS, rhetorically connected (via a discourse relation) to the utterance due to be generated, and
- no other explicitly realized referent of the same semantic type (agent, object, etc.) and with the same morphological attributes - viz. gender, number, person is introduced in a previous utterance in the current SDRS, rhetorically connected to the utterance due to be generated.

As we can see, the second condition involves the interaction between the prononinal anaphora generation component and a surface realizer modeling a certain language (via at least a morphological grammar). Further on in the paper, we give an extended example illustrating the operation of these two conditions in order to perform a filtering on anaphora generation.

3.2 Pronominal Anaphora Generation Steps

Under the pragmatic constraints previously specified, the anaphoric turn generation is performed via the following processing stages:

1. translation of the logic form expressing the meaning of the utterance due to be generated, into an utterance according to the syntactic pattern; in this process, the semantic entities in the syntactic pattern are customized according to the

---

$^{49}$ The labels of the rhetorical relations are equivalent to their names (Asher and Lascarides, 2003).
values of the corresponding variables in the logic form;

2. determination of the verb class for the lexical item due to instantiate the predicate in the syntactic pattern, and application of the appropriate transformation rule (cf. § 2.4);

3. backtracking of the discourse structure (SDRS) came from the rhetorical structuring component, investigating whether each entity in the syntactic pattern had already been introduced in a previous utterance (i.e., speech turn in dialogue) rhetorically connected to the current one;

4. blocking of the pronominal anaphora generation when ambiguities arise (the referent of an object has been introduced in a previous utterance, rhetorically connected to the current one, but an accessible referent of another object, having the same grammar categories - gender, number, person, etc. exists in another prior utterance in the current discourse structure);

5. application of the appropriate anaphor generation rule in the pronominal anaphora generation grammar (cf. § 2.1) for each entity already introduced (i.e., initialised) in the SDRS, via a prior utterance;

6. generation of the anaphora, in linguistic form, applying the rules in the pronominal anaphora generation grammar (cf. § 2.1);

7. adjusting of the final surface form for the anaphoric utterance, using a morpho-syntactic grammar for the language being used.

Step (1) is realized via the following transformations:

1. determination, in each logic form, of the pairs of predicates of the form: entity(\text{variable}) \land equals(\text{variable}, \text{value}) where entity is one of the following: agent, object, patient, modifier;

2. transformations of the type: entity(\text{variable}) \land equals(\text{variable}, \text{value}) \rightarrow equals(\text{ent}, \text{value});

3. by virtue of the transformations shown above, each entity \text{ent} is substitute by the corresponding value \text{value} in the syntactic pattern.

Step (2) is realized according to the discussion in § 2.4; step (3) supposes first the usage of an especially defined grammar in order to represent discourse structures.

Therefore, step (3) in the pronominal anaphora generation algorithm supposes (designating through the label \pi, the utterance due to be generated) simply the test if the entity entity occurs, initialised with the value \text{value} in an utterance labeled \pi : j < i in the SDRS came from the rhetorical structuring component. This test, leaning on the SDRS grammar, is performed in two steps:

(a) apply the test only on utterances \pi_j such that a rhetorical relation \rho exists between these utterances and utterance \pi_i (due to be generated) and, only if the test is negative,

(b) apply the test on all the utterances \pi_j in the SDRS given by the rhetorical structuring component.

Steps (5) and (6) in the pronominal anaphora generation algorithm consist in applying the processing steps in the sections specified above (in the presentation of the algorithm). In the following section, all these processing steps will be illustrated via an extended example.

Until now, we have not addressed the issue of pronominal anaphora generation for the agent or patient entities, when the agent is the speaker and the patient is the destination of the message conveyed by the speaker, or, conversely, the speaker is the patient and the destination of the message is the agent. In these two cases, searching for appropriate referents in the current discourse structure is replaced by inferences imposing the selection of certain rules in the pronominal anaphora generation grammar (cf. § 2.1). Hence, denoting by \text{emitter}(\alpha) the recipient of the utterance \alpha, we have a set of rules, such as:

\text{equals}([\text{agt}], \text{emitter}(\alpha)) \Rightarrow ([\text{agt}] \rightarrow [\text{Aagt}]) \land ([\text{Aagt}] \rightarrow \text{je}).

When the agent is neither the speaker (the emitter), nor the recipient of the utterance due to be generated, the general pronominal anaphora generation algorithm is applied (searching for valid antecedents in previous utterances).

4 Extended Pronominal Anaphora Generation Example

For the text of an utterance due to be generated, taking into account the class of the verb materializ-
ing its predicate, as well as the degree of illocutionary force (Caelen and Xuereb, 2007), we find the appropriate set of rules (as specified in previous sections) to apply on the syntactic pattern.

Let us assume that we have the following dialogue situation ($\pi_i$ represents the label of the $i$-th utterance in dialogue, $U$ denotes the user and $M$, the computer):

$U$: $\pi_1$: Madame Jeanne a besoin du livre ‘X’.
(Mrs. Jeanne wants the book ‘X’.)

$U$: $\pi_2$: Elle voudrait aussi le DVD ‘Y’.
(She would like also the DVD ‘Y’.)

Here, the utterances $\pi_1$ and $\pi_2$ are connected via the (monologic) rhetorical relation $\text{Parallel}(\pi_1, \pi_2)$ in the current SDRS; in this moment, $M$ is supposed to generate an utterance due to find out when Mrs. Jeanne needs the book ‘X’, therefore the communicative intention of the utterance (labeled $\pi_3$) to be produced by $M$ can be expressed, in first-order logic (Popescu et al., 2007), as:

$$
\exists X, Y, Z: \text{object}(X) \land \text{equals}(X, \text{‘livre’}) \land \text{patient}(Y) \land \text{equals}(Y, \text{‘Jeanne’}) \land \text{need}(Y, X, Z) \land \text{good_time}(Z) \land \text{equals}(Z, t_1) \land \text{equals}(Z, t_2).
$$

Utterance $\pi_1$ is thus supposed to be a question so that any answer to it elaborates on utterance $\pi_1$; hence, the current discourse structure will be updated (Popescu et al., 2007) connecting $\pi_3$ to $\pi_1$ via the (dialogic) rhetorical relation $\text{Elab}_q(\pi_1, \pi_3)$. However, no rhetorical relation is supposed to connect $\pi_2$ and $\pi_3$. Therefore, the updated SDRS, containing utterance $\pi_3$ (due to be produced by $M$) is presented in Figure 1.

We apply step (1) in the pronominal anaphora generation algorithm; this logic form allows us to establish the mappings between semantic categories (agent, object, etc.) and the entities actually present in the utterance.

In order to determine the time of the verb instantiating the predicate of the utterance, we lean on the logic form expressing the communicative intention and on a specific rule relying on the discourse ontology.

Then, from the fact that, in the logic form, the modifier is not initialised (although it refers to a future moment of time - $t_i$ (Popescu et al., 2007)), one infers that $\pi_3$ will be an interrogative utterance. As for the pronominal anaphora generation filters, the following rules$^{51}$ trigger the appropriate decisions:

- $\text{Elab}_q$ exists between utterances $\pi_1$ and $\pi_3$, and the modifier in $\pi_1$ does not exist, hence the modifier in $\pi_3$ will be anaphoric:

$$
(\text{Elab}_q(\pi_1, \pi_3) \land \text{equals}(\text{[mod]}_{\pi_1}, \emptyset)) \Rightarrow ([\text{mod}]_{\pi_3} \rightarrow [\text{A}_\text{mod}]),
$$

- $\text{Elab}_q$ exists between utterances $\pi_1$ and $\pi_3$, and the patients in the two utterances are the same, hence in $\pi_3$ the patient will be anaphoric:

$$
(\text{Elab}_q(\pi_1, \pi_3) \land \text{equals}(\text{[pat]}_{\pi_1}, \text{[pat]}_{\pi_3})) \Rightarrow ([\text{pat}]_{\pi_3} \rightarrow [\text{A}_\text{pat}]),
$$

- $\text{Parallel}$ exists between $\pi_1$ and $\pi_2$, and $\text{Elab}_q$ exists between utterances $\pi_1$ and $\pi_3$, and the objects in $\pi_1$ and $\pi_2$ have the same gender, hence the anaphor generation will be blocked:

$$
(\text{Parallel}(\pi_1, \pi_2) \land \text{Elab}_q(\pi_1, \pi_3) \land (\text{[obj]}_{\pi_1} \rightarrow \text{[object]}::(\text{thing}) \land (\text{[obj]}_{\pi_2} \rightarrow \text{[object]}::(\text{thing}_2)) \land \text{equals}(\text{[object]}::(\text{thing}), (\text{gender}), (\text{object})::(\text{thing}_2)).(\text{gender})) \Rightarrow \neg([\text{obj}]_{\pi_3} \rightarrow [\text{A}_\text{obj}]).
$$

Thus, the syntactic pattern leads first to the non-interrogative utterance:

$M$: $\pi_3^\text{Q}: \text{Quand est-ce qu’il faudra le livre à Jeanne ?}
(When does Jane need the book?)$

This utterance has to be generated in anaphoric form, according to the algorithm presented in Section 3. Therefore, in step (2) of the algorithm, given that the class of the verb “falloir” instantiating the predicate is “VI 3” (impersonal verbs, where the agent does not exist and the object and the patient exist), the syntactic pattern is constrained by the appropriate rule (cf. § 2.4).

In step (3) in the pronominal anaphora generation algorithm, we investigate the set of previous utterances rhetorically connected to the current one (to be generated) and we decide (see the rules

$^{51}$ These rules are particularizations of more general rules, which contain, for instance, entities of the type $<\text{RhetRel}(\pi_i, \pi_j)>$.}
above) that the object of utterance \( \pi_3 \) cannot be generated in anaphoric form since this process is blocked by the existence of the object “DVD” that shares, in French, the same gender and number with the object “livre”. Nevertheless, the patient, “Jeanne” can be generated as anaphor, since \( \pi_3 \) has the same patient as \( \pi_1 \) and no other patient sharing the same morphological attributes exists in the current SDRS.

Thus, in step (6) in the generation algorithm, we find, in the utterance to be generated, the entities to be produced in anaphoric manner and we choose, in the pronominal anaphora generation grammar, the terminals realizing the anaphora in linguistic manner; this process is driven by a set of rules in the task ontology (Caelen and Xuereb, 2007), (Popescu et al., 2007):

\[
([\text{pat}] \rightarrow \langle \text{object} \rangle \mid [\text{agent}] \rightarrow \langle \text{thing} \rangle \rightarrow \langle \text{animated} \rangle) \land \text{equals}([\text{object}] \rightarrow \langle \text{agent} \rangle \mid [\text{object}] \rightarrow \langle \text{thing} \rangle \rightarrow \langle \text{animated} \rangle).
\]

For the example developed here, from these processing stages we obtain the final interrogative utterance:

\[
M: \pi_3: \text{Quand est-ce qu’} il lui faudra le livre ?
\]

(When does she need the book?)

5 Conclusions and Further Work

In this paper we have presented a framework for anaphoric utterance generation in a language generator for human-computer dialogue. The approach, semantically and pragmatically oriented, tries to avoid, as much as possible, pure morpho-syntactic treatments. Thus, we have specified a set of rules in an pronominal anaphora generation grammar, along with rules linking this grammar to syntactic patterns (for interrogative and non-interrogative utterances) and to the type of the verb instantiating the predicate of the utterance. Then, we have proposed and illustrated via an example, an algorithm for pronominal anaphora generation, however, using a grammar of French language. Nevertheless, this grammar is well separated from the general pronominal anaphora generation mechanism, which is thus independent, in principle, of the formalism used for modeling (from a morpho-syntactic point of view) a particular natural language. The algorithm described here is currently implemented and tested in ISO-PROLOG for a book reservation application developed using a generic task-oriented dialogue sytem.

In the near future, we should first refine the framework presented here so as to include negative utterances and then, more important, model deontic constraints induced by social relationships between users.

Acknowledgment

The author wishes to thank Jean Caelen from Laboratoire Informatique de Grenoble, for his continuous support and contributions to this research.

References


Anaphora Resolution for the ARE Competition

Gabriela Pavel, Iustin Dornescu, Alex Moruz
“Al. I. Cuza” University of Iași, Romania, Faculty of Computer Science
{pavelg, idornescu, mmoruz}@info.uaic.ro

Abstract

This paper presents the structure and characteristics of an anaphora resolution system and the steps made till now. The system is based on an existing rule-based system for tracking antecedents in English and was developed in order to be used to the ARE 2007 competition. The paper will also discuss the future directions of developing this system (the use of a statistical model, more semantic knowledge and the possibility of using machine learning procedures).

1 Introduction

One of the major problems of Computational Linguistics is anaphora resolution. If solved correctly, anaphora is useful in many discourse analysis problems like textual entailment and question/answering systems. An anaphor is a sequence of terms (NP) or a pronoun and its resolution means finding the right term that it refers to (a previous term – another NP or a pronoun referring the same entity called antecedent). All anaphors referring to one entity make a coreference chain containing them and the entity referred by them (the NP to be solved, the NPs and pronouns referring it). Anaphora resolution tries to identify all the antecedents for a given entity by using morphological, syntactic and semantic features of the words.

The Human Language Technologies (TLU) group of the Faculty of Computer Science the University “Al. I. Cuza” Iași, tried to solve this problem by creating the RARE (Robust Anaphora Resolution Engine) rule-based system (Cristea, Postolache, 2002a). The system (http://consilr.info.uaic.ro/RARE/) receives an English text annotated to POS, lemma, dependency, and returns lists of coreferential entities (it builds coreference chains). The results we obtained are comparable to other high ranked systems worldwide (a success rate of 61% and a recall of 73%).

This system was modified in two directions: first of all it was adapted for the ARE 2007 competition (first edition of Anaphora Resolution Exercise, a satellite event of the DAARC – Discourse Analysis and Anaphora Resolution Conference, where the system had the best results) by introducing new rules and including some more preprocessing steps.

The other direction was the adaptation for the Romanian language (a system having quite the same successful rate as the initial RARE system: 69%, but a smaller value for the recall: 60%). This new system needed the implementation of a NP-chucker and the addition of some rules using specific morphologic features for Romanian.

In next section we present some theoretical linguistics concepts concerning anaphora and we describe some past works related to automatic anaphora resolution systems as well as the current directions (section 2).

We present the RARE implementation (section 3) and the adaptation of this system for the ARE competition 2007 (section 4) and the obtained results (section 5).

2 What is anaphora resolution?

Anaphora resolution is important because it helps determine the “cohesion” of the discourse (Mitkov, 2002). Noun phrases (NP) make sense in the discourse when we know their referents from the real world. This picking of the right referent is called coreference, a process that applies to NP anaphors, but not to some other types of anaphors like the referring expressions (verbal ones).

The systems for automatic anaphora resolution takes into account morphological features (the identity of number, of part-of-speech etc.), syntactic features (where subjects are preferred to objects, like in Winograd’s system) or semantic features (in situations when the number is not relevant to anaphora resolution, a semantic relation like the synonymy could be very useful: an army, the soldiers can refer the same entity in the real world).
2.1 Previous works in this field

The efforts in dealing with anaphora resolution started 30 years ago (as we see can see in the short history of these works in Mitkov, 2002) with the Bobrow’s STUDENT system (1964). His system was a basic anaphora resolution system, because it solved only simple anaphors using very simple pattern matching rules – heuristics - (it could solve only anaphors that appears in the same context as their antecedents: The Russians had guns. They had several ...., where the pattern is <NP> had). We think this type of system is not useful, because it does not present real situations (NPs and theirs antecedents can appear in the same context very seldom).

Another system based on heuristics was Winograd’s SHRDLU, which examines all the possible antecedents preceding the analyzed NP. The antecedents are chosen according to their syntactic roles (eg. the subject is preferred to the object). Winograd’s system preferred the ‘focus’ elements (answers to wh- questions and to yes/no questions), because it was mainly used in a human computer interaction.

A very important implementation was Hobbs’ naïve algorithm (1976) operating on a syntactic tree, a system that is still used as a reference for evaluation results in anaphora resolution systems. Hobbs analyzes only the N-bar of each NP (the NP without determiners), a strategy that helps him distinguish between PP (prepositional phrases), where the same word refers to different entities: Mr. Smith saw a driver in his truck (his and a driver are coreferential, while in the next example they are not) and Mr. Smith saw a driver of his truck. Hobbs naïve algorithms uses the syntactic tree of the sentence and parses it in a left – to – right manner finding the antecedent (an NP or an S node) or tries to find it recursively in the previous sentence and so on.

3 Robust Anaphora Resolution Engine

RARE (Robust Anaphora Resolution Engine, Postolache, 2004) is a rule-based system for anaphora resolution, which has three levels of text analysis and four components (Cristea et al. 2002a).

On the first level is made up of the text to be analyzed (for which the engine will find the coreference chains). Entities on this level are called referential expressions (REs) and each of them contains a set of features or attributes (the first component of the engine). The second level is the level on which the REs are projected into projected structures (PS), and corresponds to the second component of the system (projection means the representation of morphological, syntactic and semantic features for the analyzed RE). The third level contains semantic information, in the form of discourse entities (the chains giving the ‘coherence’ of the text). This level corresponds to the fourth component of the model: the referential accessibility domain, which describes a set of rules for putting some limits to the antecedent’s search (a certain distance, a linear search). The third component of the model is made up of rules and heuristics for determining coreference (both pronominal and for NPs). RARE compares the current projected entity to each of the dereferential entities determined up to that point; if the score of the best match is high enough, the current PS will be attached to the referential chain of the DE, and if not, a new DE will be proposed (the candidate with the best score is chosen (Rich&LuperFoy)). RARE also allows for the postponing of decisions, until more data is collected by analyzing the text.

The workings of RARE can be described by analyzing the example below: I read the book. I recommend it to you.

The RARE solution to anaphora resolution is based on the application of several rules. A rule in RARE is based on simpler rules that test the agreement in morphological gender or number, the identity of...
lemmas, etc. In some cases, RARE also delays resolution (in which case RARE solves also cataphoras – the antecedents are the pronouns to be solved, and they are followed by the NPs they refer). RARE returns referential chains.

4.1 The initial RARE system

RARE takes into account a certain set of features such as lemma, number, part-of-speech, syntactic role and whether or not the current word is a proper male/female/family name. All these features receive values from either the pre-annotating process (the texts were annotated with a dependency parser prior to being sent to RARE) or at the beginning of the resolution process (information about proper nouns etc.).

RARE uses three types of rules: demolishing (rules invalidating candidates: Including Rule – two REs can’t be coreferential if one of them include the other: a book on the shelf cannot be coreferential to the shelf), certifying (rules accepting an antecedent: the same proper nouns refers to the same entity) and score rules (any other rule which gives some partial information for solving the anaphors). The model contains also score rules using semantic information from WordNet.

4.2 RARE adaptation for ARE

The system described in this section won the Anaphora Resolution Exercise 2007 (ARE, first edition: http://clg.wlv.ac.uk/events/ARE) part of the 6th Discourse Anaphora and Anaphora Resolution Colloquim (DAARC), Lagos, Portugal.

Tasks

The objective of the competition was to develop discourse anaphora resolution methods and evaluate them in a common environment. This edition focused on pronominal and NP coreference for English, which means that we had to solve pronominal anaphors (to find theirs antecedents) and to build the coreferential chains correctly. The competition included the following tasks: task 1 – the solving of pronominal anaphora on pre-annotated text (marked NPs), evaluated by means of success rate; task 2 – the solving of coreferential chains on pre-annotated text (marked NPs), evaluated by means of f-measure; tasks 3 and 4 were similar to tasks 1 and 2 respectively, with the added difficulty of not having any prior annotation of the text.

Resources used and preprocessing steps

The input consisted of text from Reuters articles tagged for POS (part of speech) and functional dependency. The input for tasks 1 and 2 had the entities that had to be solved already determined; for determining entities in the input texts for tasks 3 and 4 we used a version of the Collins parser for English. Given the special characteristics of the corpus (newspaper articles), a set of new rules had to be added to the already existing engine, in order to address such issues as quotes, for example.

New rules

The new rules that we added were mainly based on morphological features, and the idea that from a list of antecedents the one closest to the anaphor can give a better average score (Proximity Rules). We also built pattern rules for pronouns (this type of pattern saying that the it/he can appear only in some specific context – taking into account proximity, number and gender given by he/she/it/they rule for RE whose hypernym is a Person - can be also used in the statistical model). In order to improve our results, we also used WordNet relations, such as synonymy and hypernymy.

Problems encountered

One of the main factors that decreases performance is the POS tagger error rate; also, the dependency relation annotation was rather flawed and the dependency relations described could not be fully relied upon. Another possible error source was due to the NP chunker. An inaccurate NP chunker is liable to greatly decrease the results for tasks 3 and 4 (for this tasks we also encountered problems because of the difference between the NPs we determined and those indicated by the organizers). This problem was also present in tasks 1 and 2, but only in trying to determine the heads of the already marked NPs. We also had to deal with quotations such as “I go on”, said the general. In this case we have a cataphora that can be solved only by delaying the resolution.

5 Evaluation results

The ARE 2007 competition was the first of a series, and, therefore, it was mostly focused on creating a framework for evaluating anaphora resolution engines.

The initial RARE evaluation is favorable, but a thorough evaluation of the engine is not possible unless large different corpora are tested. The suc-
cess rate (computed by overlapping the chains in the golden and the chains identified by the system) was a good: 61%, while the recall was of 73%.

The results for the ARE competition are given for every task (task 2 has a similar output to that of RARE – coreference chains):

- Task 1: 71.55% (baseline 33%, average 43.5%)
- Task 2: 48.32% f-measure (53.01% precision, 45.72% recall)
- Task 3: 65.45% (baseline 19.52%, average 11.65%)
- Task 4: 58.02% (baseline 50.28%, average 33.93%)

The initial RARE model and the model we adapted for the ARE competition are substantially different. In the former, rules are built to deal with general anaphors and with a literary corpus, while the latter uses context specific rules and a newspaper corpus. In the immediate future it is important to modify the winning system at ARE by creating new rules in order to make it appropriate for other types of corpora (eg. literary corpora).

6 Conclusions

Though the system was the winner of the ARE competition, several directions in which it can be improved have been put forward. The rule-based model can be kept, but with a number of modifications, in order to improve the quality of resolution process: i) designing a system for determining the weights of score rules automatically (weights are manually introduced, to give a form of salience to the rules); ii) in order to accomplish task 2 from above, it will be useful to use machine learning techniques or genetic algorithms for optimizing the weights' values. The proposed solution, thus far, is to use a perceptron (in this case, the computational size of the problem is very large, which may prove to be a hindrance).

6.1 A mixed model

The rule-based system is adequate, but limited by the rules defined for it, and costly because a new kind of corpus will require new rules. Furthermore, a large number of rules increases execution time. The proposed solution is to use a mixed model, which will statistically solve most anaphors, which will then be corrected by the use of a small number of specific rules.

In an ideal system, the rules, as well as the anaphors, should be discovered by the system itself, which learns them from a training corpus and then adds them to the engine’s third component. A mixed system should overperform both rule-based and statistical systems.

References


Postolache O. 2004. RARE – Robust Anaphora Resolution Engine, MSc thesis at the Faculty of Computer Science, University “A. I. Cuza”, Iaşi


Stuckardt R. Design and Enhanced Evaluation of a Robust Anaphor Resolution Algorithm