CONTENTS

Invited Lectures

Knowledge in Computational Linguistics

Knowledge Processing and Discovery

Knowledge in Software Engineering

Knowledge in Distributed Computing

KNOWLEDGE IN

COMPUTATIONAL LINGUISTICS

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 29–32

IDENTIFICATION OF TEMPORAL EXPRESSIONS IN THE DOMAIN OF TOURISM

ANDREA VARGA[(1)], GEORGIANA PUŞCAŞU[(2)], AND CONSTANTIN ORĂSAN[(3)]

1. Introduction

QALL-ME (Question Answering Learning technologies in a multiLingual and Multimodal Environment)[1] is an EU-funded project that aims to develop a shared infrastructure for multilingual and multimodal question answering in the domain of tourism. The purpose of the system implementing this infrastructure is to be able to answer questions about local events such as movie showtimes, directions to sites (e.g. cinemas, hotels), etc. Investigation of the domain revealed that a large number of the user questions contain temporal constraints. This paper presents the temporal annotator employed to process English questions.

2. Temporal expressions in user questions

2.1. **QALL-ME benchmark.** The QALL-ME benchmark is a collection of several thousand spoken questions in the four languages involved in the project: Italian, English, Spanish and German [1]. The benchmark was created for two purposes: to allow development of applications based on machine-learning for QA and to enable testing their performance in a controlled laboratory setting. To date, the benchmark contains 15,479 questions related to cultural events and tourism, such as accommodation, gastro, cinemas, movies, exhibitions, etc., associated with the relevant information necessary to train and test the core components of a QA system. In this paper only the English part of the QALL-ME benchmark was used, with a total of 4,501 questions.

2.2. **Types of temporal expressions.** Temporal expressions (TEs) are natural language phrases that refer directly to time. TIMEX2 [3] is the worldwide adopted standard for the annotation of temporal expressions in text, due to its coverage and level of detail. The QALL-ME consortium has also adopted the TIMEX2 standard for the purpose of temporal expression annotation in QALL-ME. Therefore, each TE present in the user questions is supposed to be annotated with the TIMEX2 tag that captures the meaning of the TE.

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. Temporal expressions; Question Answering; Temporal annotator.
This work is supported by the EU-funded project QALL-ME (FP6 IST-033860).
[1]The project's webpage is http://qallme.fbk.eu/

TEs normally denote position in time, duration or time frequency. The following classes and subclasses of TEs can be distinguished:

1. **TEs indicating time position**
   - 1.1 **Precise TEs:** are the ones for which one can confidently determine their position on the time axis.
     - 1.1.1 **Calendar dates:** include TEs denoting specific years, months, dates, decades, centuries, millennia (e.g. *14th of September*).
     - 1.1.2 **Times of day:** such expressions can specify times of the day at the level of hour, minute, second or millisecond (e.g. *10 o'clock*, *7:53 am*). They can include references to the time zone where that specific time of day is applicable to.
     - 1.1.3 **Week references:** These expressions refer to periods of time having the granularity at the week level (e.g. *next week*).
   - 1.2 **Fuzzy TEs:** are vague or have imprecise boundaries.
     - 1.2.1 **Generic references to the past, present or future:** TEs that refer in general terms to the past, present or future (e.g. *now*).
     - 1.2.2 **Seasons, parts of the year (quarters and halves):** denote certain parts of a year (e.g. *the summer*).
     - 1.2.3 **Weekends:** TEs referring to weekends (e.g. *this weekend*).
     - 1.2.4 **Fuzzy day parts:** denote parts of the day (e.g. *afternoon*).
   - 1.3 **Non-specific TEs referring to time position:** are time position TEs mentioned in generic contexts (e.g. *nowadays*).
2. **TEs capturing durations:** indicate periods of time by specifying how long something lasted.
   - 2.1 **Precise durations:** specify exactly how long something lasted (e.g. *24 hours*).
   - 2.2 **Fuzzy durations:** denote an unspecified number of temporal units included in a period of time (e.g. *weeks*).
   - 2.3 **Non-specific durations:** are durations occurring in sentences that state generalisations (e.g. *all day*).
3. **Set-denoting time expressions:** give information about the frequency of a certain event.
   - 3.1 **Precise frequency TEs:** indicate sets of times telling precisely how often something happens (e.g. *every day*).
   - 3.2 **Non-specific frequencies:** are set-denoting TEs used in generic contexts (e.g. *some nights*).

On the basis of benchmark investigation it was noticed that QALL-ME user questions do not contain the full range of possibly existing TEs defined in the TIMEX2 annotation guidelines. Therefore, not all the classes of TEs need to be tackled by a real-time QA system aiming at answering questions in the domain of tourism.

In order to identify the most frequent types of TEs encountered in the user questions, and with a view towards evaluating a temporal expression identifier on questions in the domain of tourism, a set of 1,118 randomly selected user questions from

the QALL-ME benchmark have been manually annotated according to the TIMEX2 standard.

The distribution of types of temporal expressions in the user questions is captured in Table 1.

	Time Position		**Duration**	**Frequency**
Precise	Calendar dates	86	19	13
	Times of day	81		
	Week	12		
Fuzzy	Past, Present, Future	0	1	N/A
	Seasons and parts of year	1		
	Weekends	16		
	Day parts	20		
Non-specific	43		10	0

TABLE 1. Distribution of TEs in the QALL-ME user questions

3. The Temporal Annotator in QALL-ME

The Question Answering system developed as part of the QALL-ME project requires as part of the Question Processing stage a module that adds temporal expression annotations to a user question. As stated before, the TIMEX2 standard was adopted as temporal annotation schema for this module. Besides the issue of a shared common annotation schema among all QALL-ME partners and the issue concerning the usability of the QALL-ME benchmark outside the QALL-ME project, the idea behind adopting the TIMEX2 standard in QALL-ME was also to re-use existing annotation tools capable of annotating according to the TIMEX2 standard. Such a tool is available at the University of Wolverhampton, and a brief description of the tool can be found in [2]. Since it is able to annotate all types of existing TEs, this tool is very complex, and despite its high performance, a real-time QA system would compromise performance against a faster runtime. Also, by investigating the user questions, one can easily notice that only a few types of temporal expressions are more frequently encountered in the data, and therefore by covering only a few TE classes, a simpler TE annotator would be enough for the QALL-ME system. Henceforth, a simplified version of the existing TE identifier was implemented following the design and methodology employed in the initial TE annotator. The simpler TE annotator covers only the most frequent types of temporal expressions present in the user questions, such as certain precise TEs denoting time position (e.g. some ways of expressing calendar dates, times of the day), as well as certain expressions of precise durations. It also covers some cases of fuzzy TEs, like certain expressions referring to weekends and day parts. The set-referring TEs, as well as all non-specific TEs are not covered by the simplified TE annotator.

4. Evaluation

The set of 1,118 user questions that were manually annotated according to the TIMEX2 guidelines was used as gold standard in our evaluation. Both the original

TE identifier, as well as the simplified version employed in QALL-ME were evaluated against this gold standard both in terms of identifying parts of an annotated TE, as well as at the level of identifying correctly the entire extent of the annotated TEs. If the evaluation takes into consideration both complete matches, as well as partial matches, the original TE annotator has an F-measure of 95.5%, while the simplified version achieves an F-measure of 85.2%. When looking only at the complete TEs that were correctly identified, the F-measure obtained by the original TE identifier is 88.5%, and the one achieved by the simplified TE annotator is 72.6%.

5. Conclusions and future work

This short version of the paper presented the temporal processor used by the English question answering system developed in the QALL-ME project. Due to space restrictions, no error analysis or related work were included. The full version will present the processor in more detail and will include detailed error analysis. Comparison to other relevant approaches will also be included.

References

[1] Cabrio, E., Kouylekov, M., Magnini, B., Negri, M., Hasler, L., Orasan, C., Tomas, D., Vicedo, J.L., Neumann, G. and Weber, C: The QALL-ME benchmark: a Multilingual Resource of Annotated Spoken Requests for Question Answering. Proceedings of LREC-2008 (2008)

[2] Puscasu, G.: A Framework for Temporal Resolution. Proceedings of the LREC-2004 (2004)

[3] Ferro, L., Gerber, L., Mani, I., Sundheim, B. and Wilson G.: TIDES 2005 Standard for the Annotation of Temporal Expressions (2005)

[1] Research Group in Computational Linguistics, University of Wolverhampton, United Kingdom

E-mail address: andrea.varga@wlv.ac.uk

[2] Research Group in Computational Linguistics, University of Wolverhampton, United Kingdom

E-mail address: georgie@wlv.ac.uk

[3] Research Group in Computational Linguistics, University of Wolverhampton, United Kingdom

E-mail address: c.orasan@wlv.ac.uk

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 33–36

SEGMENTING TEXT BY LEXICAL CHAINS DISTRIBUTION

DOINA TATAR[(1)], EMMA TAMAIANU-MORITA[(2)], AND GABRIELA CZIBULA[(3)]

Abstract. Lexical chains represent a very powerful base for segmentation and many papers are devoted to this problem. In our study we apply a new method for obtaining the boundaries, a method that relies on the number of chains which end in a sentence, which begin in the following sentence and which traverse these two successive sentences. The method is successfully experimented and evaluated on ten texts from DUC02 conference both for a set of automated obtained lexical chains and for two sets of human obtained lexical chains (pair to pair and against to the manual segmentation).

1. Introduction

The purpose of linear segmentation is to obtain groups of successive sentences which are linked to each other from a specified point of view. The segmentation is often a valuable stage in many natural language applications.

The direction used for linear segmentation in this paper relies on the cohesion of a text, regarded as "a device for sticking together different parts of the text" [1]. The most usual and appropriate way of identifying cohesion for automatization is represented by lexical chains (LCs). LCs are sequences of words which are in a lexical cohesion relation with each other and they tend to indicate portions of a text that form semantic units [4], [3]; they could serve, further, as a basis for segmentation and/or summarization. Usually LCs are constructed in a bottom-up manner ([1]). The top-down way of building LCs is introduced in one of our own earlier paper [8]and we called it CTT (Cohesion TextTiling, similar with the name TextTiling introduced in [2]).

The main goal of this paper is to introduce a new method for segmentation, Lexical Chains Distribution method (LCD). The method is applied to the automated obtained LCs and to the human defined LCs for ten documents from DUC2002 competition. The segmentations obtained are compared pair to pair and against to the manual segmentation.

The paper is structured as follows: Section 2 contains the usual bottom-up method and our top-down methods for LCs building. Section 3 presents the new

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. Lexical Chains, Text segmentation, Word Sense Disambiguation, Text summarization.

LCD method of segmentation. The experiment and its results are presented in Section 4. We finish the paper with conclusions and further work directions in Section 5.

2. Building lexical chains

Usually a lexical chain is obtained in a bottom-up fashion, by taking each candidate word of a text, and finding an appropriate relation offered by a thesaurus (as Rodget [3] or WordNet [1, 6]). If this relation is found, the word is inserted with the appropriate sense in the current chain, and the senses of the other words in the chain are updated. If no chain is found, then a new chain is initiated. The following relations are used in identifying related terms in a LC (denoted as WordNet relations): synonymy, antonymy, hyperonymy, hyponymy, holonymy, and meronymy.

The algorithms to compute LCs suffer either from a lack of accuracy in WSD or from computational inefficiency. An alternative is to firstly disambiguate all text and secondly build LCs. A lexical chain obtained after disambiguating a text is represented as a sequence of the form: $w\#n(S_i), w\#n(S_j), ...w'\#m(S_k), ...$, where $w\#n(S_i)$ represents the word w disambiguated with the WN sense n in the sentence S_i and S_j and $w'\#m(S_k)$ represents the word w' disambiguated with the WN sense m in the sentence S_k. Here $w'\#m$ and $w\#n$ are related by a WN relation as described above.

To disambiguate a text we use in this paper our CHAD algorithm presented in [7, 8]. This is a Lesk's type algorithm based on WordNet. The base of the algorithm is the disambiguation of a triplet of words, using Dice's , Overlap or Jaccard's measures. In short, CHAD disambiguates at a time a triplet $w_{i-2}w_{i-1}w_i$, where the first two words are already associated with the best senses and the disambiguation of the third word depends on the disambiguations of the first two words . For CTT algorithm of determining LCs [8] we used the following observation. Due to the brevity of definitions in WordNet (WN) it is possible that the overlaps between fixed definitions (senses) of the words w_{i-2}, w_{i-1} and all definitions (senses) of the w_i are zero. The first sense in WN for w_i is associated in this case, in a "forced" way. From the cohesion point of view, the "forced" case signals that a LC might stop, and, perhaps, a new one might begin. Scoring each sentence of a text by the number of "forced" to first WN sense words in this sentence, in the graph representing the score function for all the sentences, the local maxima of the function will represent the boundaries between LCs and, also, between segments. So, in CTT method, a segment corresponds to a LC.

LCD method establishes the segments as shown in the next section.

3. Distribution of LCs used in segmentation

Let us consider that the set of LCs is described as:

$LC_1 : [S_{i1}, S_{j1}], LC2 : [S_{i2}, S_{j2}], \cdots$ where S_{ik} represents the first sentence of the lexical chain LC_k and S_{jk} represents the last sentence of the lexical chain LC_k.

Using this information about LCs, a score of each sentence to be the last sentence of a segment could be calculated. Namely, let denote by $input(S_i)$ the number of

LCs which end in S_i, by $output(S_i)$ the number of LCs which begin in S_i , and by $during(S_i, S_{i+1})$ the number of LCs which traverse S_i (which is not a beginning for LC) and S_{i+1} (which is not an end for LC). We call this case as "strict traversal". **The score of S_i to be the last sentence of a segment** is:

$$score(S_i) = \frac{input(S_i)+output(S_{i+1})}{during(S_i,S_{i+1})}$$

The justification of the formula is: if $input(Si)$ and/or $output(S_{i+1})$ are large, there is a large chance for S_i to be the final sentence of a segment, and/or for S_{i+1} to be the first sentence of the next segment. Also, if $during(S_i, S_{i+1})$ is small, this chance is enforced, because a small number of LCs which "strict traverse" two sentences indices a small link between them. So, the bigger the $score(S_i)$ is, the bigger is the chance to have a boundary between S_i and S_{i+1}. In this way, the points of local maxima in the graph $score(S_i)$ indicates the boundaries of the text $S_1, \cdots, S_n$. The above formula improves a scoring method of [4, 6] by introducing the term $during(S_i, S_{i+1})$ and by splitting the functions of a sentence to be a final sentence or a first sentence in a segment. The positive influence of this new term is studied in Section 4.

4. Experiments

The validity of the method for three sets of LCs is proved in an experiment on ten text from DUC2002. For each text, the sets of LCs are:

- LCs manually obtained, using all types of lexical relations ;
- LCs manually obtained, using only the repetition and the synonymy ;
- LCs obtained after disambiguation of all words by CHAD;
- LCs obtained by CTT.

The segmentations obtained for the first three sets of LCs by LCD method are denoted, shortly, by Manual, Partial and (also) LCD. The segmentation obtained by CTT as at the end of Section 2 is denoted (also) by CTT. The relative precisions for segmentation methods are calculated using $WindowDiff$ measure [5]. $WindowDiff$ is an error measure which counts how many discrepancies occur between the reference and the system results:

$$WindowDiff(Hyp, Ref) = \frac{\sum_{i=1}^{N-k} |r(i,k)-h(i,k)|}{N-k}$$

Here $r(i, k)$ represents the number of boundaries of Ref(erence) segmentation contained between sentences i and $i+k$ and $h(i, k)$ represents the number of boundaries of Hyp(othesis) segmentation contained between the sentences i and $i + k$. The selected value for k is 2. The precision between the segmentations Hyp and Ref is calculated as: $Precision(Hyp, Ref) = (1 - WindowDiff(Hyp, Ref)) \times 100$.

We calculated all the precisions of segmentations compared pair to pair. The conclusion is that LCD segmentation has the average precision better than the CTT segmentation: 69.6 % vs. 55.9% comparing with Manual and 66.1% vs. 60.8% comparing with Partial. Because both LCD LCs and CTT LCs rely on the same tool of WSD, we conclude that our formula of scoring and the method of segmentation is the source of this improvement. Another conclusion is that focusing only on the

repetition and the synonymy in the manually constructed LCs does not affect too much the quality of segmentation: precision of Manual relative to Partial is 82.8%.

To evaluate how much improves the denominator $d(S_i, S_{i+1})$ the quality of the segmentation we compare a manual obtained segmentation (by experts) with two automated segmentation: one obtained with LCD method as above, the second obtained with formula of scoring with the denominator equal to 1. In this second case the influence of LCs traversing a sentence is ignored. The result of this experiment confirms the importance of these LCs. Namely, for six documents the segmentations (and also the precisions) are the same, for four documents the segmentations are improved (the precisions are increased).

The methods presented in this paper are fully implemented (in Java): for LCD and CTT method we used our own systems for WSD task, scoring and segmentation.

5. Conclusion and further work

This paper argues that the use of LCs distribution could be a powerful tool for topical segmentation. The evaluation is realized by comparing obtained segments with the manual obtained segments. We intend to study a clustering approach of topical linear segmentation. The similarity between two clusters could be simply the number of LCs with the origin in the first cluster and with the end in the second one. The obtained method will be compared with that presented in this paper.

References

[1] Barzilay, R., Elhadad, M. : Using lexical chains for Text summarization, in Mani, J. and Maybury, M. (Eds) : Advances in Automated Text Summarization, 1999, MIT Press.
[2] Hearst, M.: TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages, Computational Linguistic, 1997, pp 33–64.
[3] Morris, J., Hirst, G.: Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text, Computational Linguistics, Vol 17, Number 1, 1991, pp 21–48.
[4] Okumura, M., Honda, T.: WSD and text segmentation based on lexical cohesion, Proceedings of COLING-94, pp –755–761.
[5] Pevzner, L., Hearst, M.: A critique and improvement of an Evaluation Metric for Text segmentation, Computational Linguistics, 28(1), 2002, 19-36.
[6] Stokes, N. : Spoken and Written News Story Segmentation using Lexical Chains, Proceedings of HLT-NAACL 2003, pp. 49-54.
[7] Tatar, D., Serban, G., Mihis,A., Lupea, M., Lupsa, D., Frentiu, M.: A CHAIN dictionary method for WSD and applications, Proceedings of KEPT2007, Cluj-Napoca, 2007, pp.41–49.
[8] Tatar, D., Mihis, A., Serban, G.: Lexical Chains Cohesion Segmentation and Summarization, SYNASC 2008, Timisoara, Sept. 26-29, IeAT Technical Report, pp 99-106.

[1] Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: dtatar@cs.ubbcluj.ro

[2] Akita University, Japan
E-mail address: etamaian@yahoo.com

[3] Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: gabis@cs.ubbcluj.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 37–40

RECOVERING DIACRITICS USING WIKIPEDIA AND GOOGLE

ADRIAN IFTENE[(1)] AND DIANA TRANDABĂŢ[(1,2)]

Abstract. The paper presents a method to restore diacritics using web contexts. The system receives one or more sentences in one language and uses the Google engine to recover diacritics for the sentence words. The system accuracy is similar to the accuracy of existing systems, but the main advantage comes from fact that it uses resources and tools available for free or that are easy to obtain for other languages, leading us to believe that this approach could be valid for more languages.

1. Introduction

This paper presents a method to restore diacritics using web found contexts. The system we propose receives one or more sentences in one language and uses various searches on web pages to recover diacritics for the sentence words.

For Romanian, automatic recovery of diacritics is a real challenge due to the frequency of those characters and their significant contribution in morphological disambiguation, but also since the majority of the Romanian texts on web is partially or completely without diacritics. In order to process the texts we need to know in advance the mapping between diacritics characters and letters without diacritics, i.e. what form will the diacritics words have if written without diacritics (s for ş, t for ţ in Romanian, ae for $\ddot{a}$, ss for β in German, etc.). Our system also needs to know the Google starting page for the considered language (i.e www.google.ro for Romanian).

We tested several methods for Romanian diacritics recovery and, in order to assure time optimality for the program, we consider that a best suited method is building a database containing, for every Romanian word, all the valid Romanian words with diacritics added or removed. The starting point of this database was a collection of Romanian lemmas (nouns, verbs, adjectives, etc.). Our goal is to improve the quality of the Romanian texts extracted from Ro-Wikipedia were, from almost 80.000 files, over 43.000 are without diacritics or partially with diacritics.

2000 *Mathematics Subject Classification.* 68P20, 91F20.
Key words and phrases. Information Retrieval, Diacritics recovery, Wikipedia.

2. Methodology

The diacritics recovery problem for Romanian was discussed by [3] and [1], their methods using n-grams. The method we propose uses statistics obtained over a corpus, considering the percent of diacritics in Romanian texts, their distribution among different classes, etc., and a map with the correspondence of the diacritics characters with their non-diacritics form (Ş - S; ş - s; Ţ - T; ţ - t; Ă, Â - A; ă, â - a; etc.). In order to increase the system precision and to reduce repeated searches, we built a database containing, for each word without diacritics, all the possible forms with diacritics that the considered word could have among known correctly spelled words. The existence of this database is not mandatory and we can start from an empty database, but the algorithm precision depends on the quality (and quantity) of entries in this database. Starting from a collection of lemmas we built a database in which for every word without diacritics we associate a list with all forms with diacritics, forms that exists in language usage.

The system proposed for the recovery of diacritics for the Romanian language is presented in the next section. We believe that using this organisation of the needed resources, the system could be easily adapted for other languages. For example, our tests on German language shown similar results like on Romanian.

3. The System

The system's architecture is presented in Figure 1. The steps performed by the program are described below:

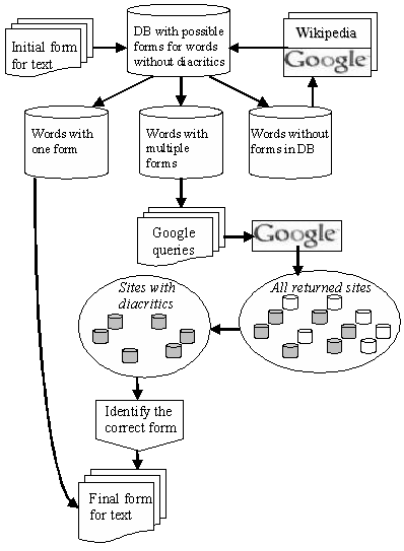


Figure 1. *System architecture*

Step 1: Initial text is split into sentences and then sentences are further split into words;

Step 2: For every word without diacritics, we search in the *database of possible forms* (DBPF) the corresponding possible value(s). If the word is in DBPF we count the number of possible forms for the word without diacritics. If there is only one correct form with diacritics, the word is directly changed to its correct form. Otherwise, if the word is not in DBPF we check the occurrences in Ro-Wikipedia and in Google. In this second case, if the frequency is high, the word is added into DBPF, else if the frequency is low, the word is considered incorrectly written and saved into a separated file with "possible problems";

Step 3: At this step we receive sentences that contain words with multiples forms in DBFP. We build a query in order to search web pages that contain similar sentences. For every sentence, the punctuation signs are removed and a query for the Google engine is built;

Step 4: We extract from web the first relevant pages returned by Google. We use therefore a crawler that uses as parameters the considered number of links returned by Google (the default value is 10), and the depth through these pages (the default value is 1);

Step 5: From downloaded sites we select only pages with texts and ignore files with images, fonts, and with configuration settings. In the selection process we identify the "correct" files with diacritics. The relevant text from all these files is concatenated in one file and the formatting tags are ignored and eliminated;

Step 6: Using the file built at Step 5 we identify the most appropriate form for words with multiple forms. We build path patterns and identify, for every word, the possible forms and its relative positions in the concatenated file;

Step 7: Context improvement: when we have several paths with similar values, we use rules for *forward* or *backward* identification. The *backward rule* searches in previous solved sentences in order to see what forms were already used for words with multiple forms. The *forward rule* puts this sentence in a waiting process until next sentences will be solved. After that we will use the identified forms in unclear situations.

Another rule can be the *maximisation rule*. This rule can be used in cases in which we have a high level of confidence in identifying the correct form for some words, and we decide to use the same form of these words in other sentences from a specified "neighbourhood".

4. System Evaluation and Discussions

In order to evaluate the system's performances, we used a large file containing the *Calimera Guidelines* (14.148 sentences). This file is transformed into a new file without diacritics by replacing the diacritics characters with their non-diacritics correspondent.

At a first run, 1.144 sentences are transformed in a deterministic way from **new form** (*without diacritics*) into **final form** (*with diacritics*) using the initial database (DBPF). In this step, sentences that contain all words with only one form in our

database are identified. At a second run, another 1.550 sentences are discovered, containing words without correspondent in the database, but words that does not contain the letters "a", "i", "s", "t" (854 new words, without diacritics), i.e. no ambiguities possible between diacritics and non-diacritics forms.

After these two runs, we discovered 6.210 words that contains the letters "a", "i", "s", "t" and without correspondence in DBPF. For these words we use Romanian Wikipedia and the Google search engine in order to find possible diacritics forms on Romanian sites.

We performed some manual tests in order to verify if a word without diacritics was transformed in the correct form with diacritics. We considered 100 sentences summing 657 words and the obtained precision is around 94.5 %. This is inferior to the precision reported in [2] of over 99 %. However, they use advanced techniques and tools of text processing for Romanian language which are not freely available, while we only use a frequency table of diacritics characters, a diacritics/non-diacritics corresponding table, a database of words with and without diacritics (which could be empty at the beginning), Wikipedia and the Google search engine.

The main errors in our system were introduced for long web documents, containing words both with and without diacritics. Envisaged solutions for these problems are: 1) the validation using diacritics percentages could be performed at paragraph level, instead at document level. 2) Saving the previous queries and creating a resource with queries and obtained solutions.

References

[1] R. Mihalcea and V. A. Năstase. An automatic method for diacritics insertion into romanian texts. *Romanian language in informational society, Expert, Bucharest*, pages 191–205, 2001.

[2] D. Tufiş and A. Ceauşu. Diacritics restoration in romanian texts. *RANLP 2007 Workshop: A Common Natural Language Processing Paradigm for Balkan Languages, September, Borovets, Bulgaria*, pages 49–55, 2007.

[3] D. Tufiş and A. Chiţu. Automatic insertion of diacritics in romanian texts. *Proceedings of the 5th International Workshop on Computational Lexicography COMPLEX, Pecs, Ungaria*, pages 185–194, 1999.

[1] "Al. I. Cuza" University of Iasi, Faculty of Computer Science, Romania
E-mail address: adiftene@info.uaic.ro

[2] Institute for Computer Science, Romanian Academy Iasi Branch, Romania
E-mail address: dtrandabat@info.uaic.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 41–44

AN APPROACH ON MULTILEVEL TEXT MINING

ADRIAN ONET[(1)]

Abstract. When different text sources are used in text mining application, a mechanism is needed to take into account the competence of each source and the validity of the knowledge provided. In this paper, we propose a method to acquire and process sets of association rules and frequent words, gathered from different sources, while at the same time taking into consideration the competence factor of each source (i.e. how trustworthy/relevant a source is). The lower level of text mining knowledge is then consolidated to form higher level of text mining knowledge. We call this a multilevel text mining process. We also consider the problem of discovering patterns from incomplete information and introduce a chase based technique to infer the missing text patterns.

1. INTRODUCTION

In the past years because of the web growth and also of other media growth there are a lot of text resources that may be mined for knowledge. Text mining applications are employed on a single text source from where the knowledge is extracted. Such systems are henceforth called level-one text mining structures. In this paper, we will consider the problem of multilevel text mining, where the text mining knowledge is collected into a knowledge base from different level-one or multilevel text mining systems. The information in the knowledge base is then consolidated to allow for another higher level analysis of this collective information. The advantage of this is that we may have multiple text mining application working on different text sources, each of text sources may be more or less relevant to our knowledge acquisition process from all text sources, in this case we may assign to each source a competence value, that is the association rules extracted from the text mining process will be sent to a higher level together with the competence of that source. The higher source that receives these association rules will be able to decide which association rules are more relevant (i.e. it comes from a very competent source or the association rules are present in most of the sources).

First let us consider a motivating example for this text mining structure: consider a medical research center that is interested on the cause related to an illness. The research center in order to collect the possible causes to that illness it searches

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. Text mining, knowledge discovery, association rule mining, frequent word set mining.

in different texts sources for different patterns related to the illness (for example it searches for association rules or words that appears often together in the same sentences). The text sources considered may be the web, different medical reports, different media sources. Each of this sources may have more or less competence, for example we may have a higher confidence in the knowledge patterns discovered from the medical reports than the one discovered from the web. The problem the medical research center faces is to check which of the given knowledge patterns discovered in different text sources has more relevance.

This paper proposes a formal method to extract knowledge from a set of knowledge sources and present the acquired knowledge in a uniform way. Information usefulness is determined based on the support value, which denotes the degree to which a certain pattern is supported by the knowledge sources, the more sources supporting a certain pattern the higher its support value. A predetermined threshold is used to define the sources that will contribute to the knowledge selection, the detail of the function which selects the most supported knowledge was described in [11]. Furthermore, we consider the problem of discovering patterns from incomplete information by inferences based on the closure of the relations used to represent the patterns. We introduce a chase based technique to infer the missing patterns. In order to represent the text mining patterns in a uniform way, the text mining patterns are mapped in ternary relations having certain properties, as explained in section 2.2. For text mining we consider only frequent word sets [10]. The representation is based on the ternary relation between two objects, as well as some pattern-specific measures. Such representation, along with a competence matrix, is then employed in the algorithm that finds most-supported information. This algorithm extends the one given in [1], making it possible to deal with sources of different competences taking into account the measures used by the sources to derive these knowledge. The extended algorithm also accepts knowledge supported by a single source only, with the condition that such a source meets the competence requirement. Moreover, the algorithm defines a threshold value, which can be adjusted to achieve optimal results (the adjustment problem is mentioned but the solution is left for further work).

The method presented in this paper distinguishes itself from previous works as it presents a solution to a unique mining strategy, where mining is carried out at multiple levels. At each level, the resulting knowledge-mining information is filtered (based on a given privacy rules) and then passed on to the next level where another mining process begins.

2. MULTILEVEL TEXT MINING

By level-one text mining structure, we refer to any current text mining structure - that is, it collects the knowledge from different text sources, rather than from pattern sources. Consider the function $M_1 : S_1 \rightarrow K$, where S_1 represents the set of all level-one text mining structures, and K represent the power set of all knowledge patterns. We will call M_1 the first level knowledge function. We need to mention that the knowledge function may not provide the entire knowledge patterns for a given source

for reasons of information privacy. Rather, it may provide only a subset of the information or even none. As an example consider S_1 the level-one text mining structure used by Source WEB, where the result of applying the first level knowledge function can be: $M_1(S_1) = \{F(\{bird - flu, pandemics\} : 10\%; \{heart, attack, chest, pain\} : 8\%); A(\{atherosclerosis \rightarrow \{heart, attack\}\} : (8\%, 65\%))\}$, where F and A represent the set of frequent itemsets and the set of association rules, respectively. We also define the competence function $C : S \rightarrow (0, 1]$, where S represents the set of all mining structures and C assigns a competence value to each of this mining structures.

Let us consider the following set $T = \{S_1, S_2, \ldots, S_n\}$, a level-two mining structure is one that has as entries the first level knowledge function M_1 restricted to T, the competence function C restricted to T, and has as output the relevant patterns discovered from T. The second level knowledge function M_2 can now be defined as the natural extension from M_1. For generalization purposes we will denote with M the knowledge function defined on any text mining structure with values in K.

An n-level text mining structure can now be defined as a mining structure that has as entries a knowledge function M restricted only to a given set of at most level-n text mining structures, and a competence function C restricted on the same set of text mining structures. The multilevel text mining problem is formulated in the following way: given the knowledge function M restricted on the set $\{S_1, S_2, \ldots, S_m\}$ and a competence function C restricted on the same set, construct a mining structure S that will find knowledge patterns from the existing patterns given by M. A solution for the multilevel text mining problem is given by the following algorithm:

Algorithm Multilevel Mining

Input: $M(S_1), M(S_2), \ldots, M(S_m), C(S_1), C(S_2), \ldots, C(S_m)$

Output: The set of most relevant patterns

(1) create uniform representation $M^u(S_1), M^u(S_2), \ldots, M^u(S_m)$
(2) restore new knowledge that may be inferred from the given incomplete knowledge patterns based on the patterns properties.
(3) given the competence of each source, and using the pattern representation from step 1, find the most supported patterns throughout the sources.
(4) consolidate the results based on each individual pattern's properties.

The first step is needed because: a) representing the knowledge in a uniform way makes it possible to relate knowledge of different patterns, b) a well-defined representation helps recover knowledge from existing patterns based on their known properties, and c) a uniform representation significantly simplifies the process of mining relevant patterns from different sources, this step is described in [12].

In the third step, the most relevant information is collected through the given patterns, taking into account source competence, pattern frequency throughout the sources, and the measures used to find a pattern by the given sources. The detailed description for this step may be found in [11]

3. CONCLUSION

In this paper, we presented an abstraction of the text mining process on multiple levels. We introduced the concept of multilevel text mining and a method to derive

the most supported knowledge patterns from sources that each has a predetermined competences value. For future work, we intend to consider sources that provide inaccurate knowledge patterns.

References

[1] S. Puuronen and V. Y. Terziyan. Knowledge Acquisition from Multiple Experts Based on Semantics of Concepts, Knowledge Acquisition, Modeling and Management (EKAW), pp. 259-273, 1999.
[2] S. Abiteboul, R. Hull, and V. Vianu. Foundation of Databases, Addison-Wesley, 1995.
[3] S. Ceri, G. Gottlob, and L. Tanca. Logic Programming and Databases, Springer-Verlag, Berlin Heidelberg, 1990.
[4] P. N. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining, Addison Wesley, 2005.
[5] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proc. of SIGMOD, pp. 207-216, June 1993,
[6] B. Omelaenko, V. Terziyan, and S. Puuronen. Multiple expert knowledge acquisition: experimental investigation of three voting strategies. In Proc. of Step'98, Human and Artificial Information Processing, Jyvaskyla, Finland, pp. 88-97, 1998.
[7] P. Lenca, B. Vaillant, and S. Lallich. Association Rule Interestingness Measures: Experimental and Theoretical Studies, Quality Measures in Data Mining (43), Springer, pp. 51-76, 2007.
[8] R. V. Book and F. Otto. String-Rewriting Systems, Springer-Verlag Berlin , 1993.
[9] D. Maier. Theory of Relational Databases, Computer Science Press, 1983.
[10] Lebart, L., Salem, A., Berry, L, Exploring Textual Data, Kluwer Academic, 1998.
[11] Onet Adrian, Higher Order Mining from Sources with Different Competences, Proceedings of World Congress on Computer Science and Information Engineering, CSIE 2009,Los Angeles, USA, IEEE Computer Society, pp. 325-330
[12] Onet Adrian, Mining from incomplete patterns, Proceedings of International Conference on New Trends in Information and Service Science, NISS 2009, Beijing, China, accepted to be published.

[1] Concordia University, Montreal, Canada
E-mail address: a_onet@cs.concordia.ca

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 45–48

NATURAL LANGUAGE BASED USER INTERFACE FOR ON-DEMAND SERVICE COMPOSITION

MARCEL CREMENE[(1)], FLORIN CLAUDIU POP[(2)], STEPHANE LAVIROTTE[(3)], AND JEAN-YVES TIGLI[(4)]

Abstract. The aim of this research is related to services on-demand creation, by assembling other existent services, using a natural language based interface. The original aspect of our approach is the use of a joint approach: semantic-based (connects the natural language with the concepts associated with the services) and pattern-based (insures that the resulted services are always reliable). The application field is related to intelligent buildings.

1. Introduction and problem statement

Service composition is a mean to create new services by composing existent services. Usually, the service composition task is solved by a human expert because it requires an understanding about the services semantics. This means that, the creation of a new service requires usually the human expert intervention, either before, by prediction, or after the user requirement was expressed. A priori service creation should be based on the prediction about all possible user request but such an approach is practically impossible. Even if the human expert will try to predict only the most probable user requests, this task will be an extremely costly one. A posteriori service composition by the human expert, after the user request is known, requires an important time, costs, and usually is not an option because the user wants the service as soon as his request was expressed.

The objective of this paper is propose such a system, that is able to create services on-demand, starting from the natural language based user request and assembling existent services according to this request. In particular, our application field concerns the intelligent environments. The services are offered, in this case, mainly by the various devices spread in the intelligent space (ex. a building) but we can also use remote web services.

2. Proposed solution

The proposed system is called Natural Language Service Composer(or NLSC). System architecture is depicted in Figure 1. This architecture is composed by two

2000 *Mathematics Subject Classification.* 68T50.
Key words and phrases. Natural Language, User Interface, Service Composition.

main parts: a) the Natural Language Processor (NLP), which is composed by a set of tools necessary for user request analyze and b) the Service Composer (SC). The Natural Language Processor transforms the user request into a machine readable, formal, request. This formal request will be used as input for the Service Composer. An existent English dictionary, WordNet[3], is used instead of creating our particular ontology. The Service Composer is based on a middleware platform called WComp [2]. This platform is targeted mainly for intelligent environment applications. WComp was designed for supporting dynamic assembling of services provided by hardware devices. Web services and UPnP services in general may be used through this platform also. The AoA (Aspects of Assembly) mechanism that comes with WComp allows the developer to create composition patterns and use them at runtime in order to modify the service architecture. The formal request (the NLP output) will be used in order to select the services and also the AoA patterns. Once we have selected the services and the patterns, the WComp platform is able to create almost instantly the new, composite service. Also, WComp platforms is responsible for devices/services discovery and dynamic architectural reconfiguration support. Contrarily to other pattern-based approaches, AoA patterns can be combined, superposed. Thus, a large number of valid combinations (services) may be created.

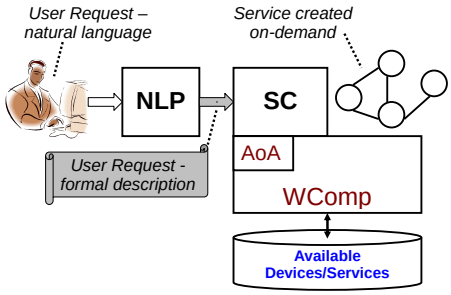


FIGURE 1. NLSC system architecture

The NLP input is a sentence that is either, written by the user or obtained from an existent voice recognition system. The sentence is decomposed in a word collection. The link words are eliminated. In order to do that, we use a list of link words. For each word, we apply a stemming procedure: the verbs are passed to infinitive, the nouns are passed to single form. Finally, we obtain a list of words that we will call further "request concepts". The semantic matching is based on what we have defined as the "conceptual distance". The conceptual distance is a measure of the similarity between two concepts. A specialized dictionary called WordNet [3] was use on this purpose.

In order to compute the semantic matching, we use a representation called "conceptual graph". The conceptual graph has as nodes the user concepts and the service concepts. The arcs connect each user concept to each service concept. The weight of each arc represents the *conceptual distance* between the concepts. In order to select

the service concepts that are similar to the user concepts, we need to apply the following two transformations to the conceptual graph: a) find the minimum distance path in the graph (include all nodes) using the Kruskal algorithm that calculates the minimum spanning tree (MST); after this transformation each service description will be connected to 2 text segments; and b) for each triplet (service concept, user concept 1, user concept 2) keep the arc that has the minimum weight. Thus, we obtain a concepts pair.

Once the services are selected according to the concepts pairs, we apply all the AoA patterns that correspond to the selected services. At this moment, only the service selection was implemented. The AoA selection (based on the user request) remains as further work.

3. Implementation

Scenario. The user request is: *"I want to use my phone to turn off the light, turn on the TV and play some music on HiFi"*. All the relevant services are identified and then composed, by applying the AoA composition patterns. The system will select, among all available service, only the services that are matching the concepts from user request. Once the services are selected, the system creates the service architecture by applying the AoA patterns. The result is depicted in the figure 2. The implementation is based on the WComp platform discussed before.

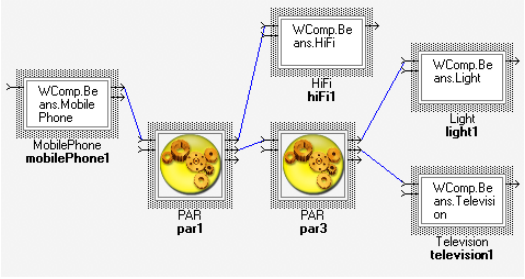


Figure 2. The dynamically composed service for Scenario 1

4. Conclusions

The original aspect of our proposal, compared with the related work (described in [1], [4] and [5] but not detailed here because of the very limited space) is the mixed approach: semantic and pattern-based. This approach combines the advantages of the both: thanks to composition patterns, it allows us to build complex composite services, that are always valid and functional. With other approaches (interface, logic, semantic based), that are not using patterns/templates, it is very difficult to create complex architectures that are are valid and work correctly. In particular, AoA patterns can be composed and this helps us to overcome the limitations of the traditional patern-based approach (not very flexible).

In the same time, the service composition is user driven, by natural language (voice) and allows the user to get the service he want on-demand. From this point of view, our solution is less restrictive than the other solutions.

Another important advantage of our solution is the reuse of WordNet [3] free dictionary, that is acting like a huge ontology. Due to this, we can relax very much the limitations for the natural language, imposed by solutions where an ontology (usually restricted) must be created by the developer. Otherwise, the creation of a rich ontology is a very costly task and our solution succeeded to avoid it. Thus, for describing the services, their developer just need to use a correct English.

References

[1] Bosca, A.; Corno, F.; Valetto, G.; Maglione, R., *On-the-fly Construction of Web services Compositions from Natural Language Requests*, JOURNAL OF SOFTWARE (JSW), ISSN : 1796-217X, Vol. 1 Issue 1, pag 53-63, July 2006.

[2] Cheung-Foo-Wo, D.; Tigli, J.-Y.; Lavirotte, S.; Riveill, M., *Self-adaptation of event driven component-oriented Middleware using Aspects of Assembly*, In 5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC), California, USA, Nov 2007.

[3] Cognitive Science Laboratory, Princeton University, *WordNet a lexical database for the English language*, Website, 2006
`http://wordnet.princeton.edu/`

[4] Fujii, K.; Suda, T., *Semantics-based dynamic service composition*, IEEE Journal on Selected Areas in Communications, Vol 23(12), pag 2361- 2372, Dec 2005.

[5] Larvet, P., *Web service with associated lexical tree*, European Patent, EP1835417.

(1) Technical University of Cluj-Napoca, str. Baritiu, no. 26, phone: 0264-401813
E-mail address: `cremene@com.utcluj.ro`

(1) Technical University of Cluj-Napoca
E-mail address: `florin.pop@com.utcluj.ro`

(3) University of Nice Sophia-Antipolis
E-mail address: `stephane@lavirotte.com`

(3) University of Nice Sophia-Antipolis
E-mail address: `tigli@polytech.unice.fr`

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 49–52

DERIVATIONAL MORPHOLOGY MECHANISMS IN AUTOMATIC LEXICAL INFORMATION ACQUISITION

S. COJOCARU, E. BOIAN, AND M. PETIC[(1)]

Abstract. The paper deals with the derivational morphology mechanisms which permit automatic acquisition of the lexical resources for the Romanian language. In this context, the possibilities of derivational sequence construction are subject to review. A separate compartiment is dedicated to formal rules of morphological derivation for some Romanian affixes. Then, the aspects of word validation are discussed. Finally, a model of automatic lexical aquisition using derivational and inflectional morphology is presented.

1. Introduction

One of the methods of lexical resources completion constitutes the generation of a word form performed by *inflection*. *Derivation* means the creation of a new word by adding some affixes to the existent lexical bases. Different distinctive features are formulated in [1]. Anyway, the discussions about the strict delimitation between inflection and derivation do not end here. Nevertheless, there are examples which may prove the relation between inflection and derivation [2].

The *aim* of the article is the studying of the derivational morphology mechanisms which permit automatic acquisition of the lexical resources for the Romanian language.

In this context, the possibilities of derivational sequence construction will be under discussion. A special section is dedicated to the formalisation of rules of derivation for some Romanian affixes. Then, the aspects of automatic generated derivatives validation are discussed. Finally, a model of automatic lexical acquisition using derivational and inflectional morphology is presented.

2. The word structure and its derivational sequence

Taking into consideration the ideea of the model FAVR [3] and the affix classification [4], a word is an entity consisting of three types of fundamental entities: *header* (combination of prefixes), *root* and *ending* (lexical suffixes and/or a grammatical suffix). The generic structure of the word can be described through the following regular

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. derivational morphology, derivation, prefix, suffix, lexical resources.

expression that uses the usual agreements (in adition, the zero suffix is considered to be a grammatical suffix):

*(simple prefix)*root (lexical simple suffix)*(grammatical suffix)* [3].

The roots together with affixes form a set of *morphemes*, which can be defined as: the smallest units that convey meaning in the structure of a language [5]. In the process of derivation it is important the order in which the affixes have been added (with the possible vocalic and/or consonant alternations). For example, the word *antimuncitoresc*, which is valid for Romanian language, consists of a simple prefix *anti-*, a root (a) *munci*, (in engl. (to) *work*) and 2 simple lexical suffixes *-tor* and *-esc*. In order to establish which of the derivatives can be considered valid, we verified the presence of the derivated words in the existent electronic documents on Internet. The derivational sequence is:

$[munci]_{Vb} \to [muncitor]_{Nn} \to [muncitoresc]_{Adj} \to [antimuncitoresc]_{Adj}$.

The derivation sequence from above is the one that has no alternatives. In the case of the derivative *descentralizator* the situation is different. The derivative consists of a simple prefix (*des-*), a root (*central*) and 2 lexical simple suffixes (*-iza* and *-tor*). In order to establish which of the derivatives can be considered valid we proceeded in the same way, as in the example described above, by checking the presence of derivatives in the existent electronic document on Internet. It is clear that it is possible to form 2 derivatives from the verb *centraliza*: *centralizator* (in engl. *centralizing*) and *descentraliza* (in engl. *decentralize*). So, the derivation sequence has 2 variants, namely:

$[central]_{Adj} \to [centraliza]_{Vb} \to [centralizator]_{Nn} \to [descentralizator]_{Nn}$,
$[central]_{Adj} \to [centraliza]_{Vb} \to [descentraliza]_{Vb} \to [descentralizator]_{Nn}$.

3. Formal rules for derivation

Besides the problem of derivatives analysis there is a wish to have the possibility to generate new derivatives, taking into acount the stem and affix peculiarities [4].

Thus we can formulate the following rule for the derivatives with the prefix *re-* and suffix *-re*. Let ω be the infinitive of a verb, then $\omega\prime = re\omega re$ is a noun, for example: (a) citi $\to$ *re*citi*re* (in engl. (to) read $\to$ *re*reading).

Another known affix is the prefix *ne-*. Thus, let ω be an adjective of the form $\omega\prime = \omega\beta$, where $\beta \in$ { *-tor, -bil, -os, -at, -it, -ut, -ind, -înd*}, it will be generated derivatives of the form $\omega'' =$ ne $\omega\beta$, the result will be also an adjective, for example: lucrător $\to$ *ne*lucrător (in engl. working $\to$ non-working). In the process of derivation with the prefixes *ne-*, *re-* and suffix *-re* vocalic and/or consonant alternations are not observed.

So an interest appears in the process of derivation with the suffixes *-tor* and *-bil*. Both of them have the same origin. Let ω be the infinitive of the verb of the form $\omega\prime = \omega\beta$, where $\beta \in \{-a, i\}$, then it is possible to form the derivatives of the form $\omega'' = \omega\beta\gamma$, where $\gamma \in \{-tor, -bil\}$ is adjective/noun. These rules are too general and they will be specified later.

This examination includes the verbal lexical simple suffix *-iza* which has a very strong relation with the lexical suffixes *-ism* and *-ist*. Let ω be an adjective/noun of

the form $\omega\prime = \omega\beta\gamma$, where $\gamma \in \{-ism, -ist\}$. Then it is possible to say about the derivatives in the case:

(1) $\beta \in \{-an, -ian\}$, then the word $\omega\prime\beta = \omega iza$ is verb, for example: rom*an* → roman*iza* (in engl. Roman → (to) Romanize);
(2) $\beta \in \{-ean\}$, then the word of the form $\omega\prime e\boxed{\text{a}}n = \omega iza$ is a verb, where $\boxed{\text{a}}$ represents the cut out of the vowel a, for example: europ*ean* → europen*iza* (in engl. European → (to) Europeanize);
(3) $\beta = \mu ic$, where:
   - $\mu \in \{-at, -et, -ot, -if\}$, then the word $\omega\prime = \omega\mu iza$ is verb, for example: dram*atic* → dramat*iza* (in engl. dramatic → (to) dramatize).
   - $\mu \notin \{-at, -et, -ot, -if\}$, then the word $\omega\prime = \omega\beta iza$ is verb, for example: clas*ic* → clasic*iza* (in engl. classic → to get a classic feature);
(4) $\beta \in \{-ura\}$, then the word of the form $\omega\prime = \omega ur\boxed{\text{a}}iza$ is verb, for example: caricat*ură* → caricatur*iza* (in engl. caricature → (to) caricature).

4. The new word validation

One of the methods of new word validation consists of manual verification of every new generated derivative in order to check the correspondence to the semantic and morphologic rules. But specific disadvantages of a manual work, namely considerable time resources and possibility to mistake appear even if the procedure is performed by a specialist in domain.

Another method of validation consists in verification of the derivatives in the electronic documents on Internet. The searching on Internet is made for the documents typed only in Romanian language. In addition, it is necessary to consider: the possibility to exclude the word segmentation, the part of speech of the derivatives, the meaning of the found word and of the generated derivative [4]. Taking into account that the words on the Web pages can contain spelling mistakes, the solution will be the use of reliable resources, for example existent corpora.

5. The concept of the derivation cycle

Following the information stated above, the lexicon completion can be realized with the help of automatic means. Starting with the derivation rules, an algorithm which forms a set of words which correspond to the derivation constraints is going to be elaborated. This algorithm of derivation is applied to these words and the result is a set of derivaties. Therefore not all the derivatives correspond to the norms of human language. After applying the method of validation, we obtain correct words on the basis of language. These words are inflected with the help of the programs of inflection [6] that result in a set of inflected words. This very set can complete the initial lexicon, making it actual.

Nevertheless after a cycle of bringing the lexicon up to date it is possible to apply another similar cycle. So, after a finite number of cycles it is likely to finish the process of completion, in the end obtaining a "filled" (saturated) lexicon which will be complete from the point of view of derivation.

6. Conclusions

The derivational rules formalisation for some Romanian affixes offer the possibility to elaborate algorithms for the lexical resources completion. The process of new derivatives validation is one that raises many questions and it seems that there are solutions though there are some difficulties in this process. Thus, it is impossible to neglect the aspect of source credibility in the process of word validation. In this context the word validation using the existent corpora seems to be the best solution.

The automatic completion cycle model for lexical resources by the derivational and inflectional mechanisms allow the consciousness of the steps in the process of lexicon enrichment.

References

[1] Booij, G. Inflection and derivation In: Keith Brown (Editor in Chief) Encyclopedia of Language and Linguistics. Second Edition, Volume 5, pp. 654-661. Oxford: Elsevier, 2006.

[2] Between Inflection and Derivation: Paradigmatic Lexical Functions in Morphological Databases. Em: East West Encounter: second international conference on Meaning - Text Theory, Moscow, Russia, August, 2005.

[3] Tufiş, D., Diaconu L., Barbu A. M., Diaconu C. Morfologia limbii române, o resurs lingvistică reversibilă şi reutilizabilă, Limbaj şi Tehnologie, Editura Academiei Române, Bucureşti, 1996, pp. 59-65.

[4] Petic, M. Completarea automată a resurselor lingvistice româneşti, Lucrările atelierului "Resurse lingvistice şi instrumente pentru prelucrarea limbii române", Ed. Universităţii "Al. I. Cuza", Iaşi, România, 2008, pp. 151-160.

[5] A Computational Model of Derivational Morphology, Dissertation zur Erlangung des akademishen Grades des Dr. Rer. Nat. An der Universitat Hamburg eingereich von Bruce Mayo Konstantz, den 4. Ooktober, 1999.

[6] Boian, E., Ciubotaru, C., Cojocaru, S., Colesnicov, A., Demidova, V., Malahova, L. Lexical resources for Romanian. Scientific Memoirs of the Romanian Academy, ser.IV, vol.XXXVI, Bucureşti, România, 2005, pp. 267-278.

[1] Institute of Mathematics and Computer Science, 5 Academiei str. Chişinău, MD-2028, Moldova

E-mail address: sveta@math.md, lena@math.md, mirsha@math.md

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 53–56

NAMED ENTITY RECOGNITION FOR ROMANIAN (RONER)

LUCIAN MIHAI MACHISON [1]

Abstract. Named Entity Recognition (NER) is the process of automatic identification of selected types of entities in a free text. This article focuses on the extraction of three kinds of entities - proper names, location names and organization names - from Romanian texts. The system implemented for this task relies on the GATE framework, and includes a series of tools developed and configured specifically for the RoNER system, such as gazetteers and a JAPE (Java Annotations Pattern Engine) transducer.

Named Entity Recognition (NER) is the process of automatic identification of selected types of entities in a free text. The entities are proper name of persons, location names and organization names. The Named Entity Recognition is a part of the greater task of Information Extraction, together with the Event Extraction [1]. Some examples of named entities are: proper person names (e.g. "John Smith"), locations (e.g. "Cluj", "Romania") and organizations (e.g. "Parlament", "Universitatea Babes-Bolyai").

1. Existing Systems in the Named Entity Recognition Domain

1.1. **Identifying Unknown Proper Names in Newswire Text.**

The first system studied was "Identifying Unknown Proper Names in Newswire Text" presented in [2]. This system considers as the entities of interest people, products, organizations and locations. It aims to identify these entities automatically from large corpora, without relying only on listings of name elements. It performed good on the task of name entity recognition, it's highest results are Precision 88% and Recall 73%.

1.2. **FASTUS: A Finite-state Processor for Information Extraction from Real-world Text.**

Another system developed for NER task is FASTUS. The description of this system is presented in [3]. FASTUS is a system which relies on a nondeterministic finite-state language model that produces a phrasal decomposition of a sentence into noun groups, verb groups and particles. After this, another finite-state machine recognizes domain-specific phrases based on combinations of the heads of the constituents found in the first pass. The FASTUS system performed good, with 44% Recall and

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. Named, entity, recognition, gazetteer, JAPE, transducer, GATE.

55% Precision on a blind test of 100 texts, which was among the best scores in the evaluation.

2. RoNER Structure

The main component used is the GATE system [4]. It functions as a framework which supports its own components as well as new components designed by the user and which can be then added to the system. For the development of RoNER several components were added to the GATE system: three gazetteers developed for Romanian and a JAPE transducer, also developed for the Romanian language. The gazetteers provide the system with a list of entities (person names, locations, organizations) and the JAPE (Java Annotations Pattern Engine) transducer provides a set of rules for matching specific patterns in the text aimed at identifying the named entities which appear in the text.

2.1. **The GATE System.**

The processing resources for the GATE program are packaged in a collection called ANNIE (A Nearly-New Information Extraction System), but any of them can be used individually. ANNIE consists of the following main processing resources: tokeniser, sentence splitter, POS-tagger, gazetteer, finite state transducer, orthomatcher and coreference solver. Another resource, which is also used in the RoNER system, is the semantic tagger, consisting of hand-crafted rules written in JAPE (Java Annotations Pattern Engine) format, which describes patterns to match and annotations to be created as a result. It also provides finite state transduction over annotations based on regular expressions.

2.2. **The Gazetteer.**

The gazetteer is a component of the GATE system which is concerned with the recognition of the named entities by comparing tokens from the text with entities in a list. These lists have been previously generated using my programs and contain some proper names, locations and organizations. If the token from the text matches an item from one of those lists it is a very high probability that the token belongs to the same category as the item in the list.

For the Romanian language there are no such lists of names, locations and organizations. In order to do this, I developed some programs which takes as an input a Romanian corpus, extracts from that corpus named entities and places them in the appropriate kind of list. Every program for generating a gazetteer list is based on some files which contain information about possible previous and successor words for named entities (names, locations, organizations). Also, for locations and organizations, a list of possible first words could be created. Such words enter in the composition of the named entity and they provide a very accurate way of placing the entity in the proper list. After the splitting into words is done, a function is applied to each of the words which determines whether the word should be included or not in the gazetteer list. The function calculates a score based on the words surrounding it, the position in the text and the fact that the word is capitalized. After running the programs on a 300 files corpus, the results were three lists, one for each named entity

type. Only the unique items, with a number of appearances above a certain threshold were kept. These list will be included in the GATE system, in the Gazetteer section.

2.3. **Modifying the GATE Components.**

For the Romanian NER system, there are three GATE components used. The first one is a Tokeniser which has the purpose of splitting the text of a given document into tokens. The resulted document is then given to a Gazetteer, which uses lists of names, locations and organizations in order to recognize and automatically annotate some of the patterns and help the use of the third component involved. The third component is a Transducer and it has the purpose of applying certain rules on the document in order to discover and annotate named entities.

In order to proper use the Gazetteer and the Transducer, certain resources had to be translated or adapted for the Romanian language. In the Gazetteer, there are resources (lists of words), which are used to identify certain words, which can then be used in the Transducer as clues for finding the named entities. The lists that were generated by the use of the Gazetteer programs were also included in these lists of words.

Another change which had to be performed was the one made to the ANNIE Transducer, where rules had to be modified in order to suit the needs for the Romanian language. In GATE, the rules are written in special files of type JAPE (Java Annotations Pattern Engine). A JAPE grammar consists of a set of phases, each of which consists of a set of patterns or action rules. The phases run sequentially. Rules have the general format: "LHS $--> $ RHS", where LHS is the lists of conditions in the left side of the rule and the RHS are the annotations to be performed.

After all the tools have been run (Tokeniser, Gazetteer and Transducer) on the document, the annotations will be made for each of the named entity category. These annotations will be visible in the document window of the GATE system.

3. RoNER Evaluation

The RoNER evaluation was made using Precision and Recall functions. In order to evaluate the Romanian Named Entity Recognition system, its performance had to be compared to that of a human annotator. That is why I have created and then manually annotated a corpus of 10 files. I have chosen for my corpus to be a collection of articles from the Romanian newspaper "Adevarul", because it has to be a collection of "natural texts". The RoNER system was run on it and the results were manually compared. Then, for every document the Precision, Recall and F-Measures and also the average value for each of these measures were calculated. The average values were:

Person: Precision:27; Recall:43; F-Measure:32
Location: Precision:75; Recall:67; F-Measure:66
Organization: Precision:74; Recall:52; F-Measure:58

4. Conclusion and Further Work

The results obtained by this system were not very high. The highest performance was recorded when annotating Location entities, followed by performance for the Organization and then the lowest performance scored by the recognition of Person entities. This result was expectable, because the tagging of person names is a harder task that the recognition of organization names, which in turn is harder that the recognition of location names (because of the context).

The RoNER system could be improved by increasing the size and the quality of the gazetteers, by writing a completely new Transducer with rules developed especially for the Romanian language and by adding additional rules to the current Transducer.

The purpose of this project was to develop a Romanian Named Entity Recognition system and the development of this project succeeded. This project shows that such a system can be done and, even with limited resources, it is possible for it to achieve significant performances.

5. Bibliography

[1] The Oxford Handbook of Computation Linguistics, Edited by Ruslan Mitkov, Oxford University Press, 2003 (Chapter 30, Information Extraction, by Ralph Grishman)
[2] Identifying Unknown Proper Names in Newswire Text, Inderjeet Mani, T. Richard MacMillan, Proceedings of Workshop on Acquisition of Lexical Knowledge from Text, pp. 44-54, 1993
[3] FASTUS: A Finite-State Processor for Information Extraction from Real-world Text, Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel and Mabry Tyson, SRI International, USA, Proceedings of IJCAI-93, Washington, DC, USA, August 1993
[4] GATE: an Architecture for Development of Robust HLT Applications, Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Department of Computer Science, University of Sheffield, UK, Proceedings of ACL, 2002

[1] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: mihai_machison@yahoo.com

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 57–60

WEB INTERFACE FOR ROUGE AUTOMATIC SUMMARY EVALUATOR

ALEXANDRU RARES ZEHAN[(1)]

Abstract. The exponential growth of available information over the last years has introduced the need of some kind of summarization to remove the redundant or less important pieces and to keep only the essence. At first the text summarization was human-made, followed by automatically summarization techniques. Based on these facts, a set of metrics (e.g. measures) were developed to evaluate the summaries extracted. For automatic evaluating summaries, using different metrics, it was developed a tool called ROUGE (Recall-Oriented Understudy for Gisting Evaluation). While ROUGE must be present on every machine where the evaluation takes place, in this paper, we suggest a way for bringing this tool to the web for centralizing and simplifying the process for different kind of users to benefit from the automatic text summarization evaluation.

1. Automatic text summarization

Text summarization is a task of producing shorter text from the source, while keeping the information content in the source, and summaries are the results of such task [1]. These summaries can be classified into two categories: indicative and informative. Indicative summaries are the kind of summaries from which the user can make use before referring to the text (e.g. to judge relevance of the source text). On the other hand the user can make use of the informative summaries in place of the source text. Another classification of the summaries based on how they are composed: extracts and abstracts. Extracts summaries are the kind of summaries formed by extracting the most important sentences from the source text, while the abstracts can contain newly produced text [2].

The main goal of this paper is to present an idea for simplifying the process of using an automatic summaries evaluation tool (e.g. ROUGE, described with more details in next section), by users from different domains of activity. Section 3 contains more details about one way to implementation this goal.

2. Automatic summarization evaluation

Metrics that can be used to accurately evaluate the various appropriateness to summarization are needed. The simplest and probably the ideal way of evaluating

2000 *Mathematics Subject Classification.* 68T50, 03B65, 68U35.
Key words and phrases. summarization, automatic evaluation, ROUGE, web.

automatic summarization is to have human subjects read the summaries and evaluate them in terms of the appropriateness of summarization. However, this type of evaluation is too expensive for comparing the efficiencies of many different approaches precisely and repeatedly. There were needed automatic evaluation metrics to numerically validate the efficiency of various approaches repeatedly and consistently. Automatic summaries can be evaluated by comparing them with manual summaries generated by humans. The similarities between the targets and the automatically processed results provide metrics indicating the extent to which the task was accomplished. The similarity that can better reflect subjective judgments is a better metric.

ROUGE (Recall-Oriented Understudy for Gisting), the most frequently used automated summary evaluation package [4], is closely modeled after BLEU for MT evaluation [5]. It includes measures for automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. The measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans [6]. ROUGE introduces in addition four different measures: ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S and a ROUGE-S extension, called ROUGE-SU.

ROUGE-N (N-gram co-occurrence statistics) is an n-gram recall between a candidate summary and a set of reference summaries. ROUGE-N is computed as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in \{\text{Reference Summaries}\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{\text{Reference Summaries}\}} \sum_{gram_n \in S} Count(gram_n)}$$

Where n stands for the length of the n-gram, $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. ROUGE-L (Longest common subsequence) computes the ratio between the lengths of the two summaries LCS and the length of the reference summary. To improve the basic LCS method, ROUGE designers introduced another metric called ROUGE-W (Weighted longest common subsequence) or weighted longest common sub-sequence that favors LCS with consecutive matches. ROUGE-W can be computed efficiently using dynamic programming. Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. ROUGE-S (Skip-Bigram co-occurrence statistics) measures the overlap ratio of skip-bigrams between a candidate summary and a set of reference summaries. One potential problem for ROUGE-S is that it does not give any credit to a candidate sentence if the sentence does not have any word pair co-occurring with its references. ROUGE-SU (an extension of ROUGE-S) adds unigram as counting unit.

3. Web interface for ROUGE

From the technical point of view, for using ROUGE, one has to know a few things about some of the computer science fields like: operating systems, different file types

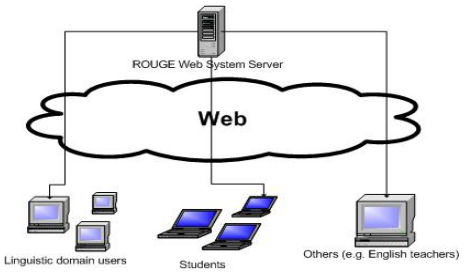


FIGURE 1. Conceptual architecture of the system

and their structures (e.g. XML file), and different platforms (e.g. Perl). These things are not very interesting (nor easy/fast to learn), if you want only to make some tests on different summaries, automatically generated or manually written.

Based on these facts, to simplify the process of using ROUGE, by users from different domains of activity, this paper introduces the idea to bring ROUGE to the Web. This thing can be achieved by creating a simple and intuitive web interface, an API for accessing ROUGE and deploy them on a machine. In this way we have a single centralized system to work with ROUGE and access it using only the browser. Figure 1 illustrates the interactions of users with this kind of system.

We implemented a simple web interface to evaluate summaries using ROUGE, but only with a small set of ROUGE's features. Using this interface, one can log in to the system and create its own evaluation project. This project consists of peers summaries (the summaries we want to evaluate), and a set of model summaries (the summaries against the evaluation is done). These summaries can be given by writing a text in a textbox, or by browsing to a file on personal file system. Before evaluating summaries, the project can be customized (Figure 2) by adding/removing ROUGE options, or by adding/removing summaries from the evaluation.

4. Conclusions and further work

Simplifying the process of using automatic evaluation tools, like ROUGE, we estimate a growth in the number of users, who can be more attracted to this kind of automatic evaluation. A next step for this system would be integration with an automatic summarization tool. An example of such a tool can be the open source project OST (Open Text Summarizer), found at http://libots.sourceforge.net.

References

[1] Fukusima, T. and Okumura, M. 2001. Text Summarization Challenge: Text Summarization Evaluation at NTCIR Workshop2. Tokyo, Japan, 2001.
[2] Mani, I. and Maybury, M., Advances in Automatic Text Summarization, MIT Press, 1999.
[3] Edmundson, H., New methods in automatic abstracting, Journal of ACM, 16(2): 264-285.
[4] Chin-Yew Lin and E. H. Hovy. 2003, Automatic evaluation of summaries using n-gram co-occurrence statistics. In Proceedings HLTN-AACL conference. Edmonton, Canada.

FIGURE 2. Demo interface

[5] K. Papineni, S. Roukos, T.Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings ACL.
[6] Chin-Yew Lin ROUGE: A Package for Automatic Evaluation of Summaries. In proceedings of workshop on text summarization branches out, Post-conference workshop of ACL 2004, Barcelona, Spain.

[1] FACULTY OF MATHEMATICS AND COMPUTER SCIENCE, BABEŞ-BOLYAI UNIVERSITY, CLUJ-NAPOCA, ROMANIA
E-mail address: zehan.rares@gmail.com

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 61–64

FEATURE SELECTION IN TEXT CATEGORIZATION USING ℓ_1-REGULARIZED SVMS

ZSOLT MINIER[1]

Abstract. Text categorization is an important task in the efficient handling of a large volume of documents. An important step in solving this task is the removal of certain features of the text that is not necessary for high precision classification. An interesting and well-founded method of feature selection are embedded methods that work directly on the hypothesis space of the machine learning algorithms used for classification. One such method is ℓ_1-regularization that is used in conjunction with support vector machines. We study the effect of this method on precision of classiffying the 20 Newsgroups document corpus and compare it with the χ^2 statistic feature selection method that is considered one of the best methods for feature selection in text categorization. Our findings show, that the ℓ_1-regularization method performs about the same as the χ^2 statistic method.

1. Introduction

Text categorization is important in information retrieval, the field that lays the theoretical foundations of search engines. As the number of pages published on the internet is growing fast, there arises a need to categorize the pages in order to facilitate further information extraction.

Most modern text categorization systems are based on machine learning algorithms for supervised classification [4], which in turn have their roots in statistics. The basic building blocks of text categorization are text documents and a set of labels. The documents are assigned to possibly more than one label, this can be formalized as a function $f : D \to 2^C$ where D is the set of documents and 2^C is the superset of labels. This function is determined by a predefined set of document and label pairs (x_i, y_i) $(i = 1, \ldots, n)$ that is called the training set. The job of a text categorization system is to predict with high accuracy the label of a document that is not encountered in the training set, that is, to find the best approximation of f based on the available data.

Traditionally text categorization is viewed as the sequential composition of two separate tasks. The first task is to find a representation for text documents that can be efficiently stored. The second task is to use a machine learning algorithm

2000 *Mathematics Subject Classification.* 68P20, 68T30.

Key words and phrases. text categorization, feature selection, support vector machine, linear programming.

that can efficiently learn the representations of documents and has a good predictive performance on new documents.

It has been observed, that it is possible to remove some of the features from the representation of all documents without incurring a performance loss, thus reducing the amount of space necessary for storing the document data [6]. In the remainder of the paper we present a feature selection method based on the ℓ_1-regularized support vector machine.

2. Feature selection by ℓ_1 regularization

A support vector machine is a learning algorithm that is able to infer a decision rule from a set of training data and then by using this rule it is able to predict some properties of previously unseen data [1]. The main advantage of SVMs over similar learning algorithms is their good performance, robustness and relatively good speed.

Let us assume, that the data is given by tuples (x_i, y_i) where $x_i \in \mathbb{R}^d$ and $y_i = \pm 1$. The SVM finds the hyperplane (described by normal vector w and bias b) that separates positive and negative examples with the largest margin. Finding the hyperplane with maximal margin can be shown to be equal to the minimization of both the total loss over the training data and the complexity of the hyperplane that is measured by the norm of its normal vector:

$$(\hat{w}, \hat{b}) = \operatorname*{argmin}_{w,b} \sum_{i=1}^{n} [1 - y_i(x_i' w + b)]_+ + \lambda \|w\|_2^2$$

where λ is called the regularization coefficient. Introducing $\|w\|_2^2$ into the minimization is called regularization because this way the the separating hyperplane is less prone to overfitting the noise in the data. This is achieved by upper bounding the length of its norm, excercising some control on the number of nonzero features. To achieve minimal length, the hyperplane has to disregard some of the less representative features in order to fit the more typical ones well. This minimization problem can be solved using quadratic programming.

In [5] and [2] it is suggested that using a ℓ_1 norm for SVMs results in sparse separating hyperplanes and thus the SVM formulation is slightly modified to:

$$(\hat{w}, \hat{b}) = \operatorname*{argmin}_{w,b} \sum_{i=1}^{n} [1 - y_i(x_i' w + b)]_+ + \lambda \|w\|_1^1$$

This formulation of the SVM can be solved with linear programming. To do this, w has to be expressed with two positive vectors as $w = w^+ - w^-$ so that $|w| = w^+ + w^-$.

Then we can formulate the linear programming solution of the ℓ_1 SVM as:

$$\begin{aligned} \min \sum_{i=1}^{n} \xi_i + \lambda \sum_{j=0}^{p} (w_j^+ + w_j^-) & \\ s.t.\ y_i(b^+ - b^- + x^{'}(w^+ - w^-)) \geq 1 - \xi_i & \quad i \in \{1, \ldots, n\} \\ \xi_i \geq 0 & \quad i \in \{1, \ldots, n\} \\ w_j^+ \geq 0, \quad w_j^- \geq 0 & \quad j \in \{1, \ldots, d\} \\ b^+ \geq 0, \quad b^- \geq 0 & \end{aligned}$$

In this method, one can not explicitly set the number of variables one wants to keep, but one has some control over them by setting the appropriate λ, and experiments show that the hyperplanes are indeed very sparse.

3. Experiments

We used the 20 Newsgroups corpus for training and testing. During preprocessing stopwords are removed and stemming is performed, numbers are converted to the token "num" and special characters are deleted.

Transforming the two-class ℓ_1-SVM to multiclass is done by training classifiers for each pair of categories. To make solving that many linear programs easier, 5000 features are pre-selected with the χ^2 statistic method, among which the ℓ_1-SVM has to choose the best ones. Four different sizes were chosen for the training set, having 10, 50, 100, and 300 randomly selected documents for each category. Using the same selections, ℓ_1-SVM, χ^2, and no feature deletion was used for feature selection. The resulting documents were then learned by an ℓ_2-SVM with $\lambda = 1$ using the LIBSVM library [3]. For the studied algorithm, λ was set to be 1/3 of the number of constraints in each linear programming problem (or if there is no solution for that λ, then $\lambda = 1$), and we used the java binding of glpk to solve the LP problems. In the case of the χ^2 statistic, the number of features selected corresponded to the number of features that the ℓ_1-SVM found to be optimal. For every training set size and every feature selection algorithm 10 runs were performed. Mean and standard deviation of performance measures are shown in Table 1.

4. Conclusions

The results show, that the ℓ_1-SVM does induce a sparse model of the data, even if this model is not more efficient for categorization than the much simpler χ^2 statistic method.

It is interesting to note, that using all features, the ℓ_2-SVM achieves better precision than the feature selection methods, this is mostly due to the fact that this corpus has well balanced word distribution, and thus many features contribute to the overall precision of a classifier.

method	#d	#f	mP	mR	mBEP	mF1
ℓ_1-SVM	10	245.50±15.72	38.00±2.20%	37.61±2.13%	37.81±2.13%	37.80±2.13%
χ^2	10	245.50±15.72	40.05±1.63%	40.27±3.10%	40.16±2.22%	40.13±2.22%
full	10	6165.10±508.81	52.42±1.71%	45.58±2.81%	49.00±1.59%	48.70±1.73%
ℓ_1-SVM	50	679.40±18.84	59.61±0.59%	59.06±0.58%	59.34±0.58%	59.23±0.62%
χ^2	50	679.40±18.84	60.66±0.56%	60.23±0.60%	60.44±0.57%	60.44±0.58%
full	50	16697.70±2091.07	69.47±0.80%	61.86±1.92%	65.66±1.11%	65.43±1.21%
ℓ_1-SVM	100	1042.40±18.81	66.75±0.70%	66.09±0.77%	66.42±0.74%	66.42±0.74%
χ^2	100	1042.40±18.81	67.05±0.34%	66.45±0.31%	66.75±0.29%	66.75±0.29%
full	100	23788.30±1387.30	74.72±0.46%	66.53±0.98%	70.63±0.47%	70.38±0.53%
ℓ_1-SVM	300	1909.10±29.83	76.64±1.67%	75.55±0.83%	75.19±1.66%	76.09±1.22%
χ^2	300	1909.10±29.83	75.32±0.24%	74.70±0.35%	75.01±0.28%	75.01±0.29%
full	300	43042.60±1227.71	81.21±0.27%	76.61±1.19%	78.91±0.67%	78.84±0.70%

TABLE 1. Results obtained for the Reuters corpus given in percentage. Notation: #d=number of documents per category, #f=number of selected features, mP=micro-precision, mR=micro-recall, mBEP=micro-breakeven, mF1=micro-F_1

ACKNOWLEDGEMENTS

We would like to acknowledge the financial support of the grant CNCSIS TD-77/2007 by the Romanian Ministry of Education and Research.

REFERENCES

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[2] P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. *Machine Learning Proceedings of the Fifteenth International Conference (ICML 98)*, pages 82–90, 1998.

[3] Chih-chung Chang and Chih-jen Lin. LIBSVM: a library for support vector machines (version 2.31), September 07 2001.

[4] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[5] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[6] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, 1997.

[1] FACULTY OF MATHEMATICS AND COMPUTER SCIENCE, BABEŞ-BOLYAI UNIVERSITY, 400084 CLUJ-NAPOCA, ROMANIA

E-mail address: minier@cs.ubbcluj.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 65–68

A ROMANIAN STEMMER

CLAUDIU SORIN IRIMIAŞ[(1)]

Abstract. This paper presents an improvement of the Romanian stemmer algorithm described on Martin Porters Snowball web-site. The changes made to the original algorithm are minimal but our experimental results indicate an increase of the accuracy with almost 10%, no loss being identified in the computational time. Two different experiments were made, the first was made on a 22,570 Romanian words vocabulary, and the second was accomplished using an article from a Romanian newspaper as input. The Romanian stemmer is based on a suffix stripping algorithm which consists of a set of rules to be applied to the input word to find its root form. Because of its efficiency, especially in regards to time and accuracy the Romanian suffix stripping algorithm is suited to be used in the information retrieval field for problems that require a smaller amount of computational time and do not necessitate that the accuracy of the result is over 80%.

1. Introduction

Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form generally a written word form. The stem need not be identical to the morphological root of the word, it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. There are several types of stemming algorithms which differ in respect to performance and accuracy and how certain stemming obstacles are overcome. Brute Force Algorithms employ a lookup table which contains relations between root forms and inflected forms. Suffix stripping algorithms do not rely on a lookup table that consists of inflected forms and root form relations. Instead, typically smaller lists of rules are stored which provide a path for the algorithm, given an input word form, to find its root form. Suffix stripping algorithms are sometimes regarded as crude given the poor performance when dealing with exceptional relations (like 'ran' and 'run'). The solutions produced by suffix stripping algorithms are limited to those lexical categories which have well known suffices with few exceptions. This, however, is a problem, as not all parts of speech have such a well formulated set of rules.

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. suffix striping, stemming algorithm, Romanian stemmer.

2. The Algorithm

The Romanian writing system uses the Latin alphabet, with five additional letters formed with diacritics: **ă, â, î, ş, ţ.** There are three specific vowels, **a, â and î** in Romanian that do not have an equivalent in English. So the following letters are vowels in Romanian: **a, ă, â, e, i, î, o, u**.

Defining R1, R2 and RV –as most of the stemmers, the Romanian stemmer uses word regions denoted by R1, R2 and RV. They are defined as follows: **R1** is the region after the first non-vowel following a vowel, or is the null region at the end of the word if there is no such non-vowel, **R2** is the region after the first non-vowel following a vowel in R1, or is the null region at the end of the word if there is no such non-vowel and **RV** is the region after the next following vowel, if the second letter is a consonant, or the region after the next consonant, if the first two letters are vowels, or the region after the third letter, otherwise (consonant-vowel case) and RV is the end of the word, if these positions cannot be found.

Example: For the word **transpunerea** the letter **n** is the first non-vowel following a vowel in beautiful, so R1 is **spunerea**. In **spunerea**, the letter **n** is the first non-vowel following a vowel, so R2 is **erea**. The second letter of the word is a consonant so RV is the region after the next following vowel **nspunerea**.

Bellow it is presented the algorithm purposed by Martin Porter, after some of the steps there exists a few notes meant to that explain the ideas for improving the algorithms accuracy. **First**, *i* and *u* between vowels are put into upper case (so that they are treated as consonants). Always do steps 0, 1, 2 and 4. (Step 3 is conditional on steps 1 and 2.)

Step 0: Removal of plurals (and other simplifications) –search for the longest among the following suffixes, and, if it is in R1, perform the action indicated.

ul, ului →delete

aua →replace with *a*

ea, ele, elor →replace with *e*

ii, iua, iei, iile, iilor, ilor →replace with *i*

ile →replace with *i* if not preceded by *ab*

atei →replace with *at*

aţie, aţia →replace with *aţi*

Step 1: Reduction of combining suffixes –search for the longest among the following suffixes, and, if it is in R1, perform the replacement action indicated. Then repeat this step until no replacement occurs.

abilitate, abilitati, abilităi, abilităţi →replace with *abil*

ibilitate →replace with *ibil*

ivitate, ivitati, ivităi, ivităţi →replace with *iv*

icitate, icitati, icităi, icitaăţi, icator, icatori, iciv, icivaă, icive, icivi, iciva, ical, icala, icale, icali, icală, →replace with *ic*

ativ, ativa, ative, ativi, ativă, aţiune, atoare, ator, atori, ătoare, ător, ători →replace with *at*

itiv, itiva, itive, itivi, itivă, iţiune, itoare, itor, itori →replace with *it*

New rule: *ator* →replace with *at*, if not prceded by *ab*

Step 2: Removal of standard suffixes –search for the longest among the following suffixes, and, if it is in R2, perform the action indicated.

at, ata, ată, ati, ate, ut, uta, ută, uti, ute, it, ita, ită, iti, ite, ic, ica, ice, ici, ica, abil, abila, abile, abili, abilă, ibil, ibila, ibile, ibili, ibilă, oasa, oasă, oase, os, osi, oşi, ant anta, ante, anti, antă, ator, atori, itate, itati, ităi, ităţi, iv, iva, ive, ivi, ivă →delete

iune, iuni →delete if preceded by *ţ*, and replace the *ţ* by *t*.

ism, isme, ist, ista, iste, isti, istă, işti →replace with *ist*

New rule: *ică*, →delete, if not prceded by *it*

Step 3: Removal of verb suffixes –search for the longest suffix in region RV among the following, and perform the action indicated. Do this step only if no suffix was removed at step 1 or step 2.

are, ere, ire, âre, ind, ând, indu, ându, eze, eascaă, ez, ezi, ează, esc, eşsti, eşte, ăsc, ăştt, ăşte, am, ai, au, eam, eai, ea, eaţi, eau, iam, iai, ia, iaţi, iau, ui, aşi, arăm, arăţi, ară, uşi, urăm, urăţi, ură, işi, irăm, irăţi, ră, âi, âşi, ârăm, ârăţi, âra, asem, aseşi, ase, aserăm, aserăţi, aseră, isem, iseşi, ise, iserăm, iserăţi, iseră, âsem, âseşi, âse, âserăm, âserăţi, âseră, usem, useşi, use, userăm, userăţi, useră →delete if preceded in RV by a consonant or *u*

ăm, aţi, em, eţi, im, iţi, âm, âţi, seşi, serăm, serăţi, seră, sei, se, sesem, seseşi, sese, seserăm, seserăţi, seseră →delete

Step 4: Removal of final vowel –search for the longest among the suffixes in R1

a, e, i, ie, ă →delete

New rules: *a* →delete if not preceded by *ş*

e →delete if not preceded by *ere* or *are*

ă →delete if not preceded by *ţ*

And finally turn *I, U* back into *i, u*.

3. Test Results

For testing the Romanian stemmer the algorithm described above was implemented in the C# programming language under Windows operating system. This implementation differs from the one available on the Snowball web-site, not just by the improvement made to the rules, but is also different from the platform point of view. On the Martin Porter web-site the implementation was done in Snowball, which is a framework for writing stemming algorithms for UNIX operating system. The accuracy of the test results was establish manually, i.e. the meaning of the words from which it resulted the same stem were compared and if the words had different sense the whole set of words was interpreted as non correct.

The first test was done on a vocabulary containing 22,570 Romanian words. The time required for stemming all the words is about 1.65625 seconds. So the average time required to stem a word is about 0.000073 seconds. From this test the estimative accuracy for the Romanian stemmer resulted to be of 80%. For the above specified vocabulary the number of words reduced was: of 4880 in step 0, 589 in step 1, 3610 in step 2, 4211 in step 3, 10033 in step 4 and 3603 words remained unchanged. Accuracy

problems were met especially at words from which the stem was a string formed by a few characters. So from different words with different meanings we have obtained the same stem. An example is given by the word abia which means just and the word abile which represents the ability of a person, the stemming algorithm will return the same stem which is the string ab. For another test we have used an article from the online version of the Romanian newspaper Ziua de Cluj. The article had about 2500 words and the accuracy for this test was noticed to be better than in case of the vocabulary, having an estimative value of 90%. The accuracy is better in this case because of the fact that in the article there were just a few words that look alike, so the cases when from two words with different meanings we obtain the same stem are reduced. The average time needed for stemming a word was about the same as in the case of the vocabulary.

4. Conclusion

With the presented algorithm for stemming Romanian words, we have obtained very good results in respect to time and more than satisfactory results in respect to the accuracy. This algorithm can be used in the information retrieval field for the problems that do not require an accuracy greater that 80%. Because of the problems encountered we can conclude that the above stemmer is not as accurate as Lemmatization algorithms could be, but it is much more time-saving. A resolution for this problem could be a hybrid algorithm that could have the suffix stripping algorithm speed and the lemmatization algorithm accuracy.

References

[1] M.F.Porter, 1980, An algorithm for suffix stripping
[2] http://snowball.tartarus.org/algorithms/romanian/stemmer.html, March 2007
[3] Dolamic, Savoy, CLEF 2007, Stemming Approaches for East European Languages
[4] Anna Tordai, Maarten de Rijke, CLEF 2005, Stemming in Hungarian
[5] David A. Hull, Gregory Grefenstette, 1996, A Detailed Analysis of English Stemming Algorithms

[1] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: sorin_irimias@yahoo.com

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 69–72

TEXTUAL ENTAILMENT AS A DIRECTIONAL RELATION REVISITED

ALPAR PERINI[1] AND DOINA TATAR[2]

Abstract. There are relatively few entailment heuristics that exploit the directional nature of the entailment relation. This paper discusses some directional methods that achieve this task. We describe our method which uses Corley and Mihalcea (2005) formula combined with the condition that must hold for the entailment to be true (Tatar et al, 2009). We have also experimented with possible derivations of this condition. We show the results that we have obtained using our applications for the RTE-1 development dataset.

1. Introduction

Recognizing textual entailment (RTE) is a key task for many natural language processing (NLP) problems. It consists in determining if an entailment relation exists between two texts: the text (T) and the hypothesis (H). The notation $T \rightarrow H$ says that the meaning of H can be inferred from T.

Even though RTE challenges lead to many approaches for finding textual entailment implemented by participating teams, only few authors exploited the directional character of the entailment relation. That is, if $T \rightarrow H$, it is less likely that the reverse $H \rightarrow T$ can also hold (Tatar et al, 2009).

This paper presents our experimenting with text entailment transformed into conditions on directional text similarities. Section 2 recalls some related work on text entailment, mainly those that exploit in some way the directional nature. In Section 3 we discuss how to implement such a directional text similarity. Section 4 contains the experimental results that we have achieved using our implementation. Section 5 presents some conclusions and possible improvements.

2. Related Work

2.1. **Probabilistic Entailment.** Glickman et al (2005) introduced a probabilistic model for determining the truth value of an entailment. In this context, the entailment relationship can be transformed into an equivalent statement about probabilities. T probabilistically entails H if T increases the likelihood of H being true. Formally $P(H = true|T) > P(H = true)$. This can be considered a directional method, since

2000 *Mathematics Subject Classification.* 68T50, 03H65.

Key words and phrases. textual entailment, directional relation, word similarity, text similarity, WordNet.

no threshold value is involved and the formal relation that is derived holds only in the direction the entailment points.

2.2. **Entailment Using Text Similarity.** The method is based on the similarity of a pair of documents defined differently depending on with respect to which text it is computed (Corley and Mihalcea, 2005). Let $sim(T,H)_T$ denote the similarity between texts T and H with respect to T. Then

$$sim(T,H)_T = \frac{\sum_{pos}\sum_{Ti\in WS^T_{pos}}(maxSim(Ti)\times idf(Ti))}{\sum_{pos}\sum_{Ti\in WS^T_{pos}} idf(Ti)} \quad (1)$$

pos involves the set of open-class words (nouns, verbs, adjectives and adverbs) in each text. The set WS^T_{pos} contains the words Ti from text T that are annotated as having the part of speech *pos*. Here $maxSim(Ti)$ denotes the highest similarity between Ti and words from H having the same part of speech as Ti. A similar formula for $sim(T,H)_H$ could be given. Based on this text-to-text similarity metric, Tatar et al (2009) have derived a textual entailment recognition system. They have demonstrated that in the case when $T \rightarrow H$ holds, the following relation will take place:

$$sim(T,H)_H > sim(T,H)_T \quad (2)$$

In Tatar et al (2007a) a simpler version for the calculus of $sim(T,H)_T$ is used: namely the only case of similarity is the identity (a symmetric relation) and/or the occurrence of a word from a text in the synset of a word in the other text (not symmetric relation).

In this paper $maxSim(Ti)$ is defined as the highest "relatedness" between Ti and words from H having the same part of speech as Ti.

3. The Proposed Method

Our approach for computing the directional text similarity is based on the formula of Corley and Mihalcea (2005). We compute a kind of "similarity" with respect to each text using (1), then we compare the resulting two "similarities" and if relation (2) holds, it is likely that we have a true entailment $T \rightarrow H$.

The main difference compared to Tatar et al (2009) in using (2) is that they used pure similarity between words in computing (1), consisting of identity and synonymy, both being symmetric. On the other hand, our approach uses a **relatedness score** for this, based on most of the WordNet relations, many of which are not symmetric. This relatedness score will be the maximum from the scores attributed to each of these relations.

Also we tried some other, more complex conditions for detecting entailment:

$$sim(T,H)_T + \sigma_1 > sim(T,H)_H > sim(T,H)_T > \theta \quad (3).$$

This means that in addition to condition (2), we not only need to reach a certain confidence level for the hypothesis based "similarity" but also we do not want to have too big difference between the two "similarities".

$$sim(T,H)_H > \theta \text{ and}$$

$$(sim(T,H)_H > sim(T,H)_T \text{ or } sim(T,H)_H + \sigma_2 > sim(T,H)_T) \quad (4).$$

Formula (4) says that, additionally, in case the difference between the two "similarities" is negligible (small σ_2), the entailment is still likely to be true.

Based on the previously discussed directional "similarity" and conditions for entailment, we have developed a Java application that recognizes textual entailment. A part of speech tagger was needed in order to distinguish the open class words. We used the Stanford POS tagger implemented in Java. For looking up words and word relations, we used WordNet, accessed through the Java interface provided by JWordNet.

We needed to compute two "similarities" based on (1). The current implementation simplifies the formula by considering $idf(w)$ to be 1 and hence the importance of the word w with respect to some documents is neglected.

We have implemented all of the entailment conditions (2), (3) and (4). We show the results in what follows.

4. Experimental Results

We have tested our application on the RTE-1 development dataset from [9].

The first run was using the entailment condition given in (2) and the resulting accuracy was 51.49%. The precision was 50.78%, meaning that it correctly recognized the entailments for a little more than half of the test pairs. The recall was 91.87%, so most of the entailment pairs were discovered.

The second run using (3) produced a 55.02% accuracy, but the parameters here need tuning. We recorded a better precision of 53.41% and lower recall of 77.48%.

The third run using (4) achieved a 55.56% accuracy. Both precision, 53.54% and recall, 82.68% were somewhat better. The threshold values were chosen empirically based on several experiments. Overall system accuracies at RTE-1 were between 50 and 60 percent [2]. The best result, 58.60%, was obtained by Glickman et al.

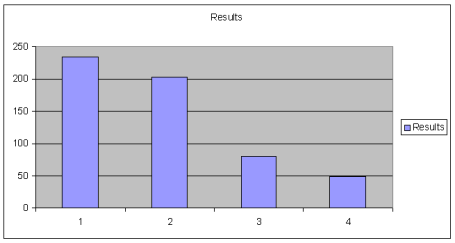


Figure 1. Resulting entailment classification: 1. True positives 2. False positives 3. True negatives 4. False negatives

5. Conclusions and Future Work

We have implemented an application that exploiting the directional nature of text similarities, computes if one text entails the other. In this application the "similarity" between a pair of words was computed using almost all WordNet relations. We

have tested our application for the RTE-1 training dataset and the best accuracy we have obtained is 55.56%. We have explored some slight changes to the condition for entailment in Tatar et al (2009).

Finally, as we have pointed out, there are improvement possibilities. Firstly, we can use a word sense disambiguation algorithm for finding the exact sense of the word to work with when computing the "similarities". Secondly, the parameters, namely σ_1, σ_2 and θ, as well as the word "similarity" scores can be further tuned, potentially using machine learning techniques.

References

[1] Corley C., Mihalcea R. 2005. *Measuring the semantic similarity of texts*. Proceeding of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor, 13-18

[2] Dagan, Glickman, Magnini. 2005. *The PASCAL Recognising Textual Entailment Challenge*. In Quinonero-Candela

[3] Glickman, Dagan, Koppel. 2005. *Web Based Probabilistic Textual Entailment*. PASCAL - Proceedings of the First Challenge Workshop, 33-36

[4] Tatar D., Serban G., Lupea M. 2007a. *Text entailment verification with text similarities*, Proceedings of KEPT2007, Cluj University Press, 33-40

[5] Tatar D., Serban G., Mihis A., Mihalcea R.. 2009. *Textual Entailment as a Directional Relation*. Journal of Research and Practice in Information Technology, Vol. 41, No. 1

[6] *Stanford POS Tagger*, http://nlp.stanford.edu/software/tagger.shtml, The Standford NLP Group. Accessed 21-Mar-2009

[7] *WordNet*, 1998 http://wordnet.princeton.edu/. Accessed 21-Mar-2009

[8] *JWordNet*, 2006, http://jwn.sourceforge.net/. Accessed 21-Mar-2009

[9] *PASCAL Recognizing Textual Entailment Challenge Development Dataset*, 2005, http://www.cs.biu.ac.il/ñlp/RTE1/Datasets/dev2.zip. Accessed 21-Mar-2009

[1] Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: pai11272@cs.ubbcluj.ro

[2] Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: dtatar@cs.ubbcluj.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 73–76

ONTOLOGICAL SOLVING OF THE TEAM BUILDING PROBLEM

ANDREEA-DIANA MIHIŞ[(1)]

Abstract. The problem of selecting the best person for a particular job is not a simple one. This problem becomes more complex when there are a lot of candidates. In this case, an application capable of selecting a small group of ideal candidates or of sorting the candidates is very useful. This kind of application can be realized easily if there is an ontological job-description database and the candidates provide a similar ontological description of their competencies.

The team building problem is a complex task, and a continuous one. It starts with the effective team constitution, and continues with the team performance improvement, from the output's point of view to the team relationships point of view [1]. Ussualy, . A part of the first team constitution, is the problem of selecting the best person for a particular job. Every person already has a professional experience that can be concretized in the last jobs description or in an ontological tree-like structure of the person's competencies. If the job is also characterized by ontology, then the problem of finding the best person for that particular job reduces to the problem of ontology matching.

But the case of team building can be also considered from another point of view. Instead of seeing the future team as a set of jobs, every job being characterized by a set of competencies with a given weight, it can be seen as a set of weighted competencies. The problem of team building could be reduced to the problem of selecting the smallest set of persons that covers the ideal set of competencies.

From another point of view, the team building problem can have as starting point a group of persons, every one with their competencies, and, by lowest cost refinement of the persons, by training, the persons will match perfectly to the team structure.
The proposed ontological description of competencies can aid in the above problems solving. It is based on the fact that every people have a limited time at the job, time in which he uses in a given weight his competencies. A base competency can be a sub-competency of a more general one and a general competency can be refined by a number of simple competencies every one in a given weight. This refinement/generalization of competencies will correspond to a tree like ontology. And because a particular job can be seen as a general competency, a job can be also refined in a set of base competencies, and so, the team building problem can be reduced to an ontology matching problem.

2000 *Mathematics Subject Classification.* 68T50, 03H65.
Key words and phrases. ontology, ontology matching, team building problem, tree, metric.

In my approach, I did not take into account the problem of this kind of ontology generation, but only the selection problem. The ontology generation can be simplified if there is available a set of base competencies, and can be facilitated by a domain based selection mechanism.

1. The ontology of competencies

Ontology is a rigorous and exhaustive organization of some knowledge domain that is usually hierarchical and contains all the relevant entities and their relations [5]. There are diffrent kind of ontologies, from the simplest ones, close to a natural language description, to the formal ones. But the most formal one have the most aplicabillity [2] [3].

From the competencies' point of view, a job or a competency can be described using two or more sub-competencies. Since the job takes a fixed period of time, in this period of time, the sub-competencies are used proportionally. If the job's standard period of time is considered to be 1, the sum of fractions of time in which a particular competency is used will be 1. Following the same principle, a sub-competency can be refined by a number of simpler competencies, every one representing a fraction of the whole denoted with 1, that corespond to the parent competency. This process repeats until, finally, base competencies are the leafs of the tree-like ontological structure (see Figures 1 and 2).

This interpretation is a particular type of ontology in which the entities are represented by competencies and the relations are weighted. In this ontology only the subordination relations are used and the sum of all weights of branches that starts from the same node is 1.

2. Ontolgy matching

There are a lot of matching techniques [4] the most popular one being based on natural language processing techniques. Such a technique uses the similarity between the words from the two ontologies to compute a ontology similarity score. In my approach, all the competency's or job's ontology has a predefined form, and so, is simple to identify de difference between two ontologies because all the base competencies belongs to the same set.

Because of the particularity of the proposed type of ontology used, the matching technique reduces to a score computation, as seen in the following example.

3. An example

In Figures 1 and 2 are two possible descriptions of the director job and of a manager job. Let assume that both are in the same domain, so already the two ontologies are 50% identical. From the point of view of entities, the two jobs are identical, but the percentages in which countable skills and communication techniques are required/used, induce a difference.

The simplest way to compare the two ontologies is from the point of view of base competences. From this point of view, a

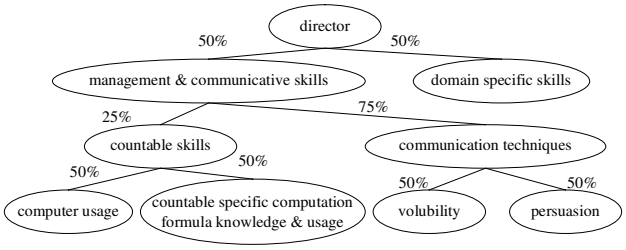


FIGURE 1. Director ontology

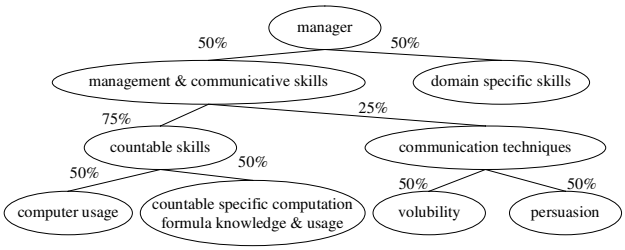


FIGURE 2. Manager ontology

$$\text{director} = 50\%(75\%(50\%\ \text{cu} + 50\%\ \text{cscfku}) + 25\%(50\%\ \text{v} + 50\%\ \text{p})) + 50\%\ \text{dss} = 0.1875\ \text{cu} + 0.1875\ \text{cscfku} + 0.0625\ \text{v} + 0.0625\ \text{p} + 0.5\ \text{dss}$$

while a

$$\text{manager} = 50\%(25\%(50\%\ \text{cu} + 50\%\ \text{cscfku}) + 75\%(50\%\ \text{v} + 50\%\ \text{p})) + 50\%\ \text{dss} = 0.0625\ \text{cu} + 0.0625\ \text{cscfku} + 0.1875\ \text{v} + 0.1875\ \text{p} + 0.5\ \text{dss}$$

We have abbreviated the competences description to every word's first letter.

In order to compare in which percentage the manager is suited for the director job, all the base competencies of the director that are covered by the manager receives the same weight as in the director (is fully covered) and the competencies that are not fully covered by the manager are weighted as in manager (not fully covered). If a competency is missing, it's score is 0.

So, manager $\rightarrow$ director $= 0.0625\ \text{cu} + 0.0625\ \text{cscfku} + 0.0625\ \text{v} + 0.0625\ \text{p} + 0.5\ \text{dss} \Rightarrow 75\%$

4. The application

The application was developed in C++ and reads the ontologies from files, from the form radix(subtree1, weight1, subtree2, weight2, ...). For instance, the director ontology appears in the following form (I have abbreviate the competences description to every word's first letter):

director(mcs(cs(cu,0.5,cscfku,0.5),0.25,ct(v,0.5,p,0.5),0.75),0.5,dss,0.5)

First step is to assure the data integrity. No base competence can be the parent of its parent and a competence cannot be refined using different competencies. Then, the base competencies are identified. If in some ontologies a non-base competence appears

as a leaf, there it will receive it's sub-tree with a medium of the basic component's weight.

Next, the application computes the scores of matching a job with another, as described in the example and finally, the person with the maximum value of the score will be selected for the target job.

In the case of team building, when the team is seen as a weighted base competencies pool, an evolutionary algorithm will be used to cover the competencies with the minimum number of persons, such that every person uses at least a given percent of his/hers capacity.

5. Conclusions

I have proposed a new ontological type competencies based description of a job and a person's competency which allows the automatization of the team-building process. In the future, I wish to automatize the process of constructing the ontologies, first assisting the construction, and next by automatically extraction from text documents or a pre-form text document, as a CV.

References

[1] Armstrong, M., A Handbook of Human Resource Management Practice, 10th edition, Kogan Page, 2006
[2] Benett, B., Fellbaum C., Formal Ontology in Information Systems, IOS Press, 2006
[3] Davies, J., Fensel, D., Van Harmelen, F., Towards the Semantic Web, John Wiley & Sons Ltd, 2002
[4] Euzenat J., Shvaiko, P., Ontology Matching, Springer, 2007
[5] http://wordnetweb.princeton.edu/perl/webwn?s=ontology

[1] Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
E-mail address: mihis@cs.ubbcluj.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 77–80

A ROMANIAN CORPUS OF TEMPORAL INFORMATION – A BASIS FOR STANDARDISATION

CORINA FORĂSCU [1]

Abstract. The paper briefly describes the main steps towards obtaining the Romanian version of the TimeBank corpus, together with the original annotations - mainly the temporal markups. The automatic import of the annotations has a success rate of 96%, whereas their preliminary evaluation shows that the transfer is perfect in 91% of the situations, it needs amendments - due mainly to the TimeML standard specific rules - in 5%, it has to be improved in 1% of the cases, due to language specific phenomena, and it is impossible for the rest, due to missing words in Romanian. These results permit an easier correspondence from the English TimeML standard to the first version of the standard applied to Romanian language - a set of guidelines for marking up Romanian text according to the ISO-TimeML language.

One of the main goals in Information Extraction is to find out *who did what to whom*, frequently continued with *when and where*. This paper intends to support the "where" part, especially for Romanian texts. After some preliminaries about temporal information in texts and its use in NLP applications, in the second section of the paper, starting with the original TimeBank 1.2. corpus (Pustejovsky et al. 2006), we briefly describe the process of creation, annotation, import and evaluation for the Romanian version of the annotated corpus. The third section presents, based on corpus evidence, the first set of guidelines for marking Romanian texts according to the TimeML language (Saurí et al. 2006). The paper ends with the main conclusions and future work on this topic.

1. Preliminaries about Temporal Information in texts

A simple sentence like *On Friday morning, after **listening** the 8 o'clock breaking **news**, Sarah will **go** to her office to **finish** the article on the last week **strikes** in Irak.* displays both explicit, and implicit temporal information: time-denoting temporal expressions (references to a calendar or clock system), and event-denoting temporal expressions (explicit/implicit/vague references to an event). This text, with the temporal elements identified, makes possible to arrange all the events on a time axis through direct or indirect temporal links. Such processings can be then used in many NLP applications (Mani et al. 2005) like: machine translation, question

2000 *Mathematics Subject Classification.* 68T50, 68T99.
Key words and phrases. corpus study, temporal information.

generation & answering, information extraction or information retrieval (tracks in competitions like TREC, CLEF, SemEval), or discourse processing.

Currently the TimeML standard is proposed as an ISO standard[1] and it integrates two annotation schemes: TIMEX2 (Ferro et al. 2005), a component technology in ACE , conceived to fill the temporal attributes for extracted relations and events, and Sheffield STAG (Setzer 2001), a fine-grained annotation scheme capturing events, times and temporal relations between them.

The TimeML standard captures the event-structure of narrative texts, and even if it has been developed mainly for English, currently there are guides for Italian, Chinese, Korean, and Spanish. The mark-up language consists of a collection of tags intended to explicitly outline the information about the events reported in a given text, as well as about their temporal relations. The standard marks:

- Events through the tags: `EVENT` - for situations that happen or occur, states or circumstances in which something obtains or holds true, and `MAKEINSTANCE` - for tracking the instances or realizations of a given event.
- Temporal anchoring of events through the tags: `TIMEX3` - for times of a day, dates - calendar dates or ranges, and durations, and `SIGNAL`- for function words indicating how temporal objects are to be co-related.
- Links between events and/or timexes through the tags: `TLINK` (Temporal) indicates 13 types of temporal relations between two temporal elements (event-event, event-timex); `ALINK` (Aspectual) marks the relationship between an aspectual event and its argument event, and `SLINK` (Subordination) marks contexts introducing relations between two events.

2. English-Romanian TimeBank parallel corpus

The TimeBank corpus consists of 183 news report documents, temporally annotated conforming to TimeML 1.2.1; it includes XML markups for document format and structure information, sentence boundary information, and named entity recognition. Even if the dimension of the corpus (4715 sentences with 10586/ 61042 unique/ total lexical units) might be too small for robust statistical learning and the annotation inconsistencies require corrections, it is considered to be the gold standard in the domain, as the proof of concept of the TimeML specifications.

The Romanian version of the corpus (Forăscu et al. 2007) was obtained through translation, with a rigorous set of guidelines. The aligned articles (4715 sentences, 65375 lexical tokens in Romanian) were double checked manually so that the next processing steps to perform better. The raw texts of the parallel corpus (sentence-level aligned through translation) were then preprocessed (tokenized, POS-tagged, lemmatized and chunked) using the TTL module (Ion 2007). For the word-level alignment we used the YAWA aligner (Tufiş et al. 2006). Two files were not aligned because of a low translation quality. All the alignments in the 181 files were manually checked, to improve the accuracy of the automatic transfer of the temporal annotations from the English version onto the Romanian one. The success rate for the import of the

[1]ISO/DIS 24617-1: *Language resource management – Semantic annotation framework (SemAF) – Part 1: Time and events.* ISO Committee Draft, Switzerland

temporal markups altogether is 96.53%. The non-transferred tags are due to missing translations (though the Romanian translation was a good and natural one), non-lexicalisations in Romanian, or missing alignments.

Preliminary evaluations (Forăscu 2008) show that the automatic temporal annotations transfer is perfect in 91% of the situations, it needs amendments - due mainly to the TimeML standard specific rules - in 5%, it has to be improved in 1% , due to language specific phenomena, and it is impossible for the rest, due to missing words in Romanian. As a side-effect of these manual evaluations, we identified and marked temporal elements (`EVENT`, `TIMEX3` and `SIGNAL`) not (yet) marked in the English corpus. Most of these new elements, if not due to inevitable manual annotation mistakes, especially for the `SIGNAL` tag, have as rationale the fact that all sentences express an `EVENT`, through their main verb. The new `TIMEX3` tags were added to vague temporal elements (*not that long ago, once*).

3. TimeML annotation standard into Romanian language

The manual evaluation and the automatic annotation import of the temporal information from English into Romanian were based on the original TimeML 1.2.1. Hence it was foreseeable to have the Romanian temporal information marked in a parallel manner with the English guidelines. When dealing with linguistic aspects of Romanian constructions, we considered the main rules of the Romanian grammar[2].

Concerning the tags `EVENT` and `MAKEINSTANCE`, the definition is the same in Romanian. The types of event-denoting expressions, as well as their extent obey the rules stated for English. During the evaluation, the TimeML rule stating that "the tag will mark only the head of a verbal phrase" imposed to modify the tag extent into Romanian in situations where adverbs were intercalated between the verbal phrase constituents (***also*** *said* - *au* ***mai*** *spus*), as well as with the reflexive verbs, when the tag was automatically transferred onto the reflexive pronoun as well (*(to) withdraw* - *(să)* ***se*** *retragă)*). One problem was with the `aspect` attribute of the `EVENT` tag, since for Romanian verbs this grammatical category is not defined the same way like for the English verbs. This is another reason why this corpus study is a step further a better definition of the "aspect" category for the Romanian verbs. For the `TIMEX3`, `SIGNAL` and `LINKs` tags the rules developed in TimeML for English are also applicable for Romanian in what concerns their definition, extent and values of their attributes.

4. Conclusions

Since the manual annotation of the temporal information is very time consuming and expensive, the import of the annotations from English is proved to be a good solution. The success rate of the import and its evaluation show that this procedure can be easily used with other types of annotations or even with other language pairs. The paper shows, based on corpus-evidence, how well the temporal theories can be applied to other languages, here with emphasis on Romanian.

[2] *The Grammar of the Romanian Language.* Romanian Academy Publishing House, 2006

Future immediate activities include finishing the evaluation and the correction/ improvement of the annotations in the parallel Romanian-English TimeBank. This will permit to clearly define the guidelines for temporal annotation in Romanian language. The parallel corpus will be useful to extract mappings between different behavior of tenses, as well as accurate translation memories. Moreover, the corpus and the annotation guidelines can be resources in future evaluation campaigns like SemEval 2010.

References

Ferro, L., L. Gerber, I. Mani, B. Sundheim, and G. Wilson (2005). *TIDES 2005 Standard for the Annotation of Temporal Expressions*.

Forăscu, C. (2008). Why Don't Romanians Have a Five O'clock Tea, Nor Halloween, but Have a Kind of Valentines Day? In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing, CICLing 2008*, Number 4919 in LNCS, pp. 73–84. Springer.

Forăscu, C., R. Ion, and D. Tufiş (2007). Semi-automatic Annotation of the Romanian TimeBank 1.2. In *Proceedings of the RANLP 2007 Workshop on Computer-aided Language Processing (CALP)*, Borovets, Bulgaria, pp. 1–7.

Ion, R. (2007). *Word Sense Disambiguation Methods Applied to English and Romanian*. Ph. D. thesis, Romanian Academy.

Mani, I., J. Pustejovsky, and R. Gaizauskas (2005). *The Language of Time: A Reader*. US: Oxford University Press.

Pustejovsky, J., M. Verhagen, R. Saurí, J. Littman, R. Gaizauskas, G. Katz, I. Mani, R. Knippen, and A. Setzer (2006). *TimeBank 1.2*. Philadelphia.

Saurí, R., J. Littman, R. Knippen, R. Gaizauskas, A. Setzer, and J. Pustejovsky (2006, January). *TimeML Annotation Guidelines, Version 1.2.1*.

Setzer, A. (2001). *Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study*. Ph. D. thesis, University of Sheffield.

Tufiş, D., R. Ion, A. Ceauşu, and D. Ştefănescu (2006). Improved Lexical Alignment by Combining Multiple Reified Alignments. In *Proceedings of the 11th Conference of EACL*, Trento, Italy, pp. 153–160.

[1] Faculty of Computer Science, University "Al. I. Cuza" of Iaşi, Romania; Research Institute in Artificial Intelligence, Romanian Academy, Bucharest

E-mail address: corinfor@info.uaic.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 81–84

COMPACTING SYNTACTIC PARSE TREES INTO ENTITY RELATIONSHIP GRAPHS

PÉTER SZILÁGYI[1]

Abstract. At the moment computers are only able to extract useful information from natural language using known patterns, reason for which they can only reach the tip of the information iceberg contained within the texts. To gain more information, shallow parsing techniques are no longer powerful enough, other methods are needed, which can execute deep parsing on the texts to determine the semantic elements and their relationships.

A new approach is introduced, with which the result of syntactic parsers can be further processed to create a graph that is much more tractable for computer programs. This goal was achieved in the following three steps: the enrollment of the words of the text into important semantic categories; the extraction of the relationships between the categorized words using syntactic rules based on the syntactic parse tree; and the disambiguation of multi-sense relationships using a probabilistic model constructed with maximum entropy.

1. Introduction

In this paper we present an approach, that takes computers one step closer to being able to deal with texts, by further transforming the syntactic parse trees into an entity–relation graph (Section 2). The processing steps are presented in detail in Section 3, after which the approach is evaluated in Section 4.

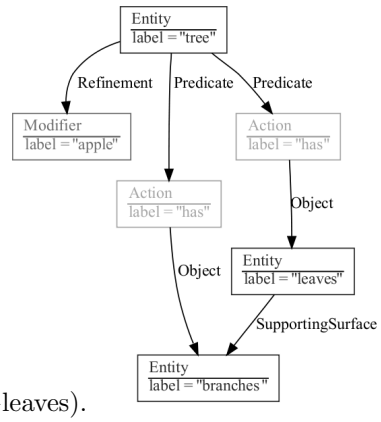
The results produced by this method are similar to those using link grammars [7], but instead of trying to figure out the best word linkage based on possible connection points, our approach was to take a parse tree and try to contract that to arrive to the final relationships. We also tried to simplify the structure as much as possible and to create a graph with very concrete elements and relationships between them, also by converting certain words into linkages, with which the link grammar does not try to deal with.

2. Entity Relation Graph

Our goal is to represent a natural text of several sentences in such a way, that removes as many formulation–related information as possible and retains only the set of entities, actions and relationships between them. To achieve this, we convert the text into an entity relation model: a directed graph where the nodes are the entities, actions and modifiers whilst the arcs represent different relations.

2000 *Mathematics Subject Classification.* 68T10, 68T30, 68T50.
Key words and phrases. syntactic parser, semantic parser, maximum entropy.

For the ease of understanding, consider the following sentence: *"An apple tree has branches and has leaves on the branches"*. This text contains three entities (tree, branches, leaves), one modifier (apple – the type of the tree) and two actions (has – the tree branches, has – the branches leaves). Amongst these elements a few relationships can be found: the basic *predicate/object* (tree→has→branches, tree→has→leaves), *refinement* (apple→tree) and a more complex relationship, *Supporting Surface* (branches→leaves).



3. Graph Construction

The syntactic parse trees (defined by the Penn Treebank project [5], introduced in [4]) of the text were used as the starting point, and further processed to reach the entity relationship graph defined previously. Our processing consists of three major steps: extraction of the semantic and metaelements; contraction of the parse tree(s); and the disambiguation of relationship senses. The first two are based on pattern matching with manually predefined rules (derived from the structure of English sentences), the third on a maximum entropy model.

The semantic and metaelement extraction is used to substitute some of the *word* nodes of the syntactic parse tree with semantic elements (entity, action, modifier) and temporary meta elements (reference, binding), which will be further processed in a later step. Entities are the subjects and objects of a sentence, and so reside in noun phrases; care must be taken for multiple entities separated by coordinating conjunctions, and for nouns preceding the main noun as they are modifiers not entities. Actions are the simplest elements to extract, all verbs in verb phrases can be converted. Modifiers can also be easily extracted since they are nouns preceding the entities in a noun phrase, used to assign a more specific meaning to the entity (these can be cumulative). References are metaelements which signal that a certain entity is not a "new" one, but one already mentioned before; these will not remain in the final graph, but will be used as connection points to other entities in the graph (or even in another sentence's entity relation graph). We handled only the simplest of the cases, when the referencing is achieved by the usage of the definite article *"the"*. Finally, bindings are metaelement space holders for special relationships, whose precise sense will only be disambiguated later; in our case the prepositions present in the text.

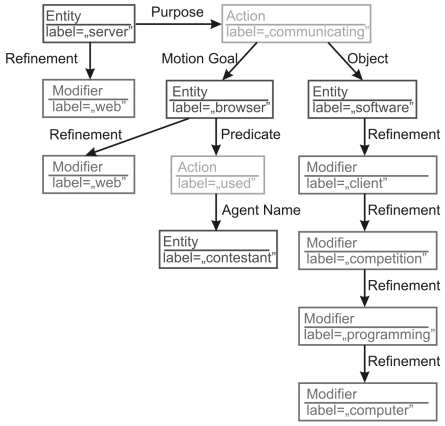
With the help of the previous step, all the *word* nodes in the syntactic parse tree should have been replaced with semantic or metaelements. In order to completely rid the entity relationship graph of text formulation constructs, the remaining internal nodes (sentence, clause, phrase) should also be removed, and the tree thus contracted into a compact graph. Sentence nodes are usually the root nodes and can be discarded. Clause nodes are more complicated, but to keep things simple, we

assumed declarative sentences only, so they can be substituted with noun phrases. Every phrase type has a dominant element (noun, verb, etc.), the place of which it holds; secondary elements on the other hand only complement the meaning of the dominant one. This way, to remove a phrase node, we first need to reconnect each secondary node from the phrase to the primary node and replace the phrase with the primary nodes (move them up one level and delete the empty parent). Lastly, we can use the reference metaelements as contraction points: the original entity node is kept, and every referencing entity is merged into it (graph arcs are reconnected to the referenced entity, and the empty referencing entity discarded).

The final major step is disambiguating the meaning of the prepositions (binding nodes), which is a hard task, as the New Oxford Dictionary of English contains 373 prepositions with a total of 847 meanings [1]. Since only contextual information can help in the disambiguation process, a probabilistic approach based on maximum entropy was taken (first presented in [8, 9]) using the corpus from the *SemEval–2007* workshop [6]. In order to keep things simple, the features used by the model were selected manually, these being: the preposition itself and the set of word stems present in the sentence, each annotated whether it is located before or after the preposition being processed. This way, the model reached a precision of 68.66% on the coarse grained evaluation and 59.89% on the fine grained one.

4. Evaluation

Our planned usage scenario for the entity relationship graph is the representation of information held within a U.S. patent, to be able to automatically check for collisions and to help verify that a company indeed has all the required rights for a system. As a proof of concept, the entity relationship graph built with our method from the sentence "*a web server for communicating computer programming competition client software to a web browser used by a contestant*", quoted from U.S. patent 6761631[2] can be seen to the right.



As with every approach, this too has certain limitations and drawbacks: the model does not contain any information related to time; the precision and correctness of the final entity relationship graph depends greatly on the initial parse tree (an error will be propagated throughout the compacting procedure); and since the substitution patterns and rules are handmade, there will always be unhandled cases.

It is important to note that the method presented was not designed to be upscaled to fully natural/spoken language, but instead it was concieved to work on a more controlled and less ambiguous language present in some legal documents (like patents).

5. Future Work

Possible improvements to the system include the creation of extraction rules to handle adjectives and adverbs (these are quite rare in our usage scenario, but still present here and there); the conversion of certain action nodes in the graph into relationship arcs (e.g. *"has"*, *"contains"*). Finally, although our system is capable of coping with a lot of constructs present in the patent system, the implementation of one that is robust enough to completely handle the complexity still requires further research and development.

References

[1] Kenneth C. Litkowski: *Digraph Analysis of Dictionary Preposition Definitions*. In proceedings of the SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation: Recent Successes and Future Directions, pages 9–16. Philadelphia, July 2002.

[2] Michael Lydon, John M. Hughes: *Apparatus and system for facilitating online coding competitions*. Retrieved April 2009, from http://www.google.com/patents?id=EVMRAAAAEBAJ

[3] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze: *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[4] Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz: *Building a Large Annotated Corpus of English: The Penn Treebank*, Computational Linguistics, volume 19, number 2, pages 313–330. June 1993.

[5] *Penn Treebank Project*. Retrieved April 2009, from http://www.cis.upenn.edu/~treebank/

[6] *SemEval–2007 Workshop*. Retrieved April 2009, from http://nlp.cs.swarthmore.edu/semeval/

[7] Daniel Sleator, Davy Temperley: *Parsing English with a Link Grammar*. Carnegie Mellon University Computer Science technical report CMU-CS-91-196, October 1991.

[8] Patrick Ye, Timothy Baldwin: *Preposition Sense Disambiguation Using Rich Semantic Features*. In proceedings of the 4th International Workshop on Semantic Evaluations (SemEval–2007), pages 241–244. Prague, June 2007.

[9] Patrick Ye, Timothy Baldwin: *Verb Sense Disambiguation Using Selectional Preferences Extracted with a State-of-the-art Semantic Role Labeler*. In proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pages 139–148. Sydney, November 2006.

[1] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: peterke@gmail.com

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 85–88

FROM DATABASES TO SEMANTIC WEB

LEON TAMBULEA [1] AND ANDREEA SABAU [2]

Abstract. The data on Web is represented using written documents, in different languages and in many formats. An application is able to perform searches on the Web based on a number of key words, but it is not able to extract knowledge from the search's results. This paper presents some results regarding the Web's transformation into a Semantic Web.

1. Introduction

In order to punctuate the evolution of the Web, different version numbers are allocated. Web 1.0 is characterized by static documents and the fact that does not offer interraction with the users. Web 2.0 is distinguished by the fact that the users' actions are dictating the content and the shape of the next document to be sent to the browser. Different sites and services using the Web as a development platform are included in this category (for example: *blogging*, *Wiki*, *Flickr*, *BitTorrents*, etc.). Web 3.0, also known as **Semantic Web** or **Inteligent Web**, is characterized by the possibility of applications to associate a meaning to various elements stored in the Internet.

2. Semantic Web

The fast growing volume of information published on the Internet determines the algorithms used to organize and to perform searches by the user's given data to be more and more complex. The currently implemented searching algorithms in the Internet are usually based on key words and may have as result many links with non-relevant data.

The idea of a Semantic Web came from the desire to have applications to perform more complex operations, which are usually done by a human user after the searching results are obtained: to generate knowledge using partial data, to associate a meaning to various data, and to draw a conclusion.

In order to perform some reasoning on the obtained data during a search, the elements stored in the Internet should represent informational units and relationships between them. In this case, the applications should be able to select, to analyze and to establish associations between such elements. These operations can be performed

2000 *Mathematics Subject Classification.* 68P05, 68P20, 68T30.
Key words and phrases. Semantic Web, knowledge.

by applications only if the informational elements have a fixed and known structure, even if they are stored in different locations. In other words, their storing format should be universal or standardized.

These elements are described on several levels [1, 2].

Level 1 is the base *level of metadata* and presents a way to specify simple semantic declarations. It describes information and resources on the Internet, as well as the relationships between them, using the RDF (Resource Description Framework) language. The information is stored in the form of *subject-predicate-object* (or *resource-property-value*) expressions. The unique identification of an element (resource or property) is accomplished by using an URI (Uniform Resource Identifier), in order to differentiate many similar elements that may exists in the Internet. The properties are used to describe an element. A property associated to a resource has a specific data type, and the set of the properties' names corresponding to a specific domain (or sub-domain) gives a possible *vocabulary* for it.

Level 2 is *the level of schemas* and offers a possibility to build a hierarchic description for a given set of properties. In order to give a unique meaning to each element for more persons or applications, a formalization of their description is necessary. This implies the definition of some terms which have associated a "semantic". The extensible knowledge representation language RDFS (RDF Schema, an extension of the RDF) is used on this level to describe classes of RDF-based resources and generalized hierarchies of classes (sub-classes or super-classes).

Level 3 is the *logic level* where complex languages can be used (for example Web Ontology Language) in order to model sophisticated knowledge.

3. RDF Triplets

A collection of RDF triplets is represented as a directed multi-graph with labeled edges. In order to exemplify the form and the content of the RDF triplets, the schema in 1 is considered.

The data represented in figure 1 can be stored in different Web pages, databases, or other kind of files (for example Excel sheets), and can be obtained by consulting or querying them. For example, the information represented by edges labeled with "a" and the connected nodes contains information about a conference, the information represented by edges labeled with "b" and the connected nodes contains organizational information about the Faculty of Mathematics and Computer Science, and the edges noted with "c" and the corresponding nodes represent the address of the same institution. Because there are some resources which are used in common, also the collections of data may be used in common and they are represented in the same oriented graph.

Each of the labeled edges drawn in figure 1 corresponds to one triplet *(r, p, v)*. The components r and p represent the resource and the property of the triplet. Their identifiers can represent different kind of data (for example: an email address for a person or a Web link for a document), which are accessible items in the Internet because they represent real addresses. Beside this, it is allowed to define within an XML document a namespace corresponding to a real address in order to simplify

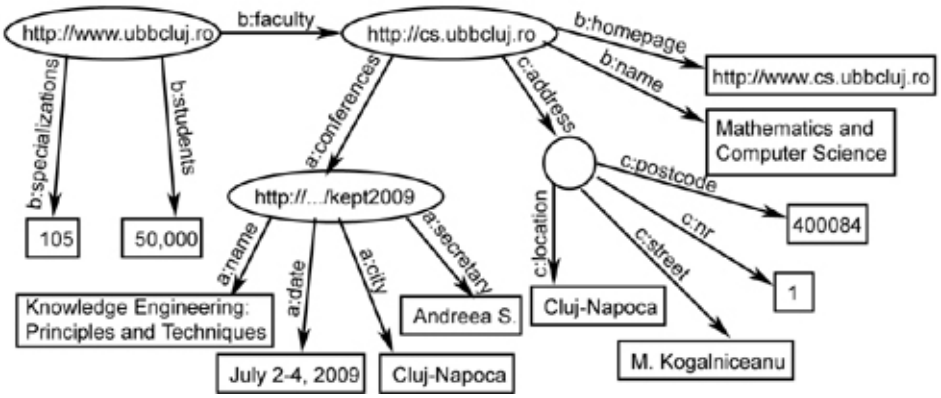


FIGURE 1. Three collections of data retrieved from different sources, but used in common in order to extract knowledge

the usage of an URI. For example, the namespace called "a" can be associated to "http://.../kept2009", in which case a unique identifier does not represent a real (accessible) address, like "a:date", "a#date", "a/date", etc.

The answer to the question "Which is the date of the Kept2009 conference?" can be determined using the collection of data noted as "b" in figure 1, even if the necessary data is not stored in the same data source. Nevertheless, finding the answer for the question "Which is the homepage of some Kept2009 conference's secretaries?" is not as easy as in the first example, because such data is not stored in any of the data sources mentioned above. In this case, some additional collections of data have to be consulted.

Different formats can be used to store *subject-predicate-object* triplets: RDF (is an XML document, having the advantage of a well-known and standard format, and also of many tools designed to manage this kind of data), Turtle (allows the RDF graphs to be written in a compact text form using specific abbreviations [6]), Notation 3 (also known as N3, offers a more compact and readable non-XML form for the RDF XML documents [5]) etc.

There is a lot of data stored in databases which could be offered (in whole or in part) for answering different queries. In order to execute such operations, a conversion from the relational model of databases in the XML model of the RDF documents should be performed. Examples of such conversions are:

- The data stored within "DBLP Computer Science Bibliography", which contains over 1 million records, is offered as XML documents. In [3] is described a method to convert this data in RDF documents (there are over 28 million triplets in present).
- Part of the Wikipedia's data is provided as RDF knowledge (over 280 million RDF triplets), according to [4].

4. Conclusions and Future Work

Semantic Web is intended to satisfy the following requirements:

- To **retrieve** data from different sources and **represent** them as RDF triplets;
- The need to have a **system to access RDF triplets** and **query collections of such triplets**;
- To create **search engines** to perform searching in RDF documents and to extract knowledge from them;
- To create **browsers for Semantic Web**.

The proposed future work includes the design of a method to formalize the transformation of the data from regular sources into data specific to Semantic Web.

References

[1] Buraga, S.C., *Considerations Regarding the Use of Semantic Web Technologies in the Context of E-business Applications*, Informatica Economica, 3(35)/2005.
[2] Herman, I., *Introduction to the Semantic Web*, 2nd European Semantic Technology Conference, Vienna, Austria, September 24, 2008.
[3] *D2R Server publishing the DBLP Bibliography Database*, http://dblp.l3s.de/d2r/.
[4] *DBpedia*, http://dbpedia.org/About.
[5] *Notation 3*, http://www.w3.org/DesignIssues/Notation3.html.
[6] *Turtle - Terse RDF Triple Language*, http://www.dajobe.org/2004/01/turtle.

[1] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, M. Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: leon@nessie.cs.ubbcluj.ro

[2] Faculty of Mathematics and Computer Science, Babeş-Bolyai University, M. Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: deiush@nessie.cs.ubbcluj.ro

KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2009
Cluj-Napoca (Romania), July 2–4, 2009, pp. 89–92

DOMAIN ONTOLOGY OF THE ROMAN ARTIFACTS FOUND IN THE TOMIS FORTRESS

CRENGUŢA MĂDĂLINA BOGDAN

Abstract. In recent years, ontologies play an important role in domains such as Semantic Web, Artificial Intelligence and Software Engineering, since they provide representations of shared conceptualizations of particular domains that can be communicated between people and applications. The present paper presents a domain ontology of the roman artifacts found in Tomis fortress of Constanta. At first, we identified and informally described the domain concepts, together with the concepts of their definition. Then we constructed the taxonomy of these concepts, using the DOLCE and D&S ontologies. Furthermore, we defined the conceptual relations between the concepts. Finally, with the assistance of the RacerPro reasoner system, we checked the consistency of the ontology. Now, we are using this ontology in order to construct virtual scenes of a software-authoring tool for virtual environments.

1. Introduction

In recent years, ontologies play an important role in domains such as Semantic Web, Artificial Intelligence and Software Engineering, mainly for the knowledge management.

An ontology is a formal specification of the concepts intension and the intensional relationships that can exist between concepts. According to Guarino's definition, "*an ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world*" [3].

Nowadays, there are some top-level ontologies (such as DOLCE, SUMO and BFO) which describe very general concepts like space, time, matter, object, event, etc., i.e. independent concepts by a particular domain or problem. Among these, we used the DOLCE ontology [4] and one of its modules D&S [1]. DOLCE is an ontology of particulars, in the sense that its domain of discourse is restricted to particulars. Other top-level ontologies might be used.

In this paper, we present an ontology of the roman artifacts found in the Tomis fortress from Constanta. To our knowledge, an ontology of the roman objects has not been constructed until now.

2000 *Mathematics Subject Classification.* 68T30, 68Q55.
Key words and phrases. historical domain, concept, ontology, taxonomy, DOLCE, Protégé.

Concept name	Informal description
amphora	Big dimensions Greek vessel of conic or cylindrical shape, with round or sharp bottom, narrow neck and two symmetrical handles; made of wood or ground; used for hold liquids (oil or wine), solid matters, or for decoration.
chiton	Classical Greek piece of garment consists of a rectangle piece of cloth, which was draped around the body and caught by an edge and shoulders with fibula.
tiara	A crown-like jeweled headdress

TABLE 1. Informal description for some important artifacts

2. The Used Methodology

The methodology of ontology construction is based on the few existent methodologies, like ontology development 101 and other ones. From these methodologies, we used the method that is presented in [5]. According to this method, in order to construct an ontology we follow the next steps: a) determine the domain and scope of the ontology; b) consider reusing existing ontologies; c) enumerate important terms in the ontology; d) define the classes and the class hierarchy; e) define the properties and relations of classes; f) create instances.

3. Construction of an Ontology of the Roman Artifacts

3.1. **Domain and the Scope of the Ontology Identification.** The ontology models the roman epoch of the Tomis fortress-Constanta, Romania, between the years 46 A.C. and 610 A.C. and the founded objects from that period. We consider ships, vessels, constructions types, pieces, as well as clothing accessories or armament elements of the roman fighters.

3.2. **Identification of the Essential Concepts and Taxonomy Construction.** Furthermore, we identified the important concepts for the studied domain. In Table 1 we give the informal description of the semantics for some of the concepts used in the roman epoch of the Tomis fortress. The next step is the definition of the class taxonomy, in which we map each concept, identified in the previous step, in a class and sequent generalization of them. The generalization was done on the basis of the subsumption relation [4]. Furthermore, we made an ontological commitment by using of the DOLCE and D&S ontologies.

In the next, we present the taxonomy of the roman objects after the categories that represent "roots" of their sub-taxonomies. In addition, these categories are directly related by a DOLCE or D&S category. We further mention that in the construction of the taxonomy we used the OWL-DL (Web Ontology Language-Description Logic) language [8] and the Protégé editor [2]. For instance, the vessels taxonomy is presented in the figure 1.

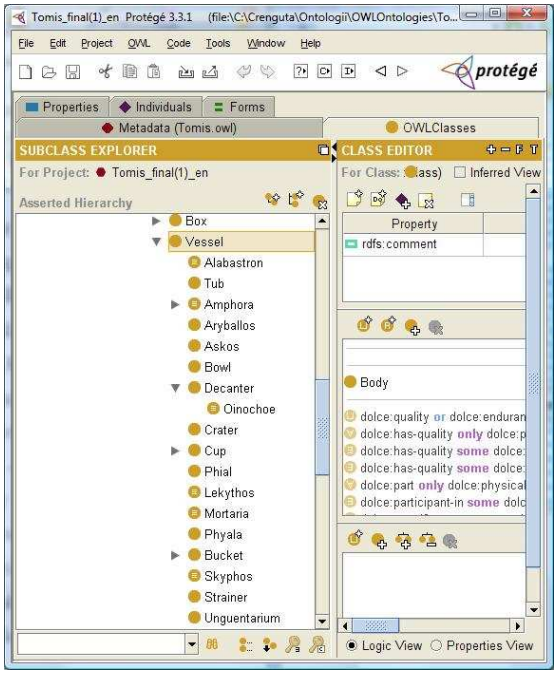


FIGURE 1. The taxonomy of vessels categories

3.3. **Identification of the Conceptual Relations.** The most of the conceptual relations of our ontology map the relations found in the DOLCE and D&S ontologies, such as *inherent-in*, *part-of*, *participate-in*, *generically-dependent-on* and so on. For instance, between the amphora and ground concepts exists the DOLCE *generic-constituent-of* relation, because there are amphorae that are made of ground. There are also relations which are specific to our ontology such as *support*, *protect*, *is-put-on*, and so on. For instance, continuing the above example, between the amphora and oil, wine or solid matter concepts there is the *hold* relation.

4. Ontology Verification and Validation

In DOLCE, the restrictions are given using a subset of the first-order logic and their verification is a long time task. That is why, we translated our ontology in OWL DL language [8] and we checked its consistency with the help of the Protégé tool [2] and the RacerPro reasoner system [7]. Furthermore, the ontology has been validated by the National History and Archeology Museum of Constanta.

5. Conclusion

We presented in this paper an ontology of the roman artifacts found in the Tomis fortress from Constanta. The taxonomy of this ontology was used for the construction

of 3D models in virtual reality [6]. Now, we are using this ontology in order to construct virtual scenes of a software-authoring tool for virtual environments.

5.1. **Acknowledgements.** This work is partially funded within the TOMIS project, no: 11-041/2007, by the National Centre of Programs Management, PNCDI-2 Partnerships program. We are grateful to Valentina Voinea for her help into the ontology validation, as well as to Andreea Matei for her efforts in the construction of this ontology.

References

[1] A. Gangemi, P. Mika, Understanding the Semantic Web through Descriptions and Situations. In Proceedings of the International Conference ODBASE03, Italy, Springer, 2003.
[2] J. Gennari, M. Musen, R. Fergerson, W. Grosso, M. Crubzy, H. Eriksson, N. Noy, S. Tu, The evolution of Protégé-2000: An environment for knowledge-based systems development. International Journal of Human-Computer Studies, 58(1):89123, 2003.
[3] N. Guarino, Formal Ontology and Information System. In Proceedings of FOIS'98, Trento, Italy, IOS Press, 1998.
[4] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, WonderWeb Deliverable D18. Ontology Library. IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web, 2003.
[5] N.F. Noy, D. McGuinness, A Guide to building ontologies: Ontology Development 101. A Guide to Creating Your First Ontology, March, 2001 at http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html
[6] Popovici D.M., Bogdan C.M., Matei A., Voinea V., Popovici N., Virtual Heritage Reconstruction based on an Ontological Description of the Artifacts, Int. J. of Computers, Communications & Control, Vol III (2008), Suppl. Issue: Proceedings of the International Conference of Computers, Communications and Control (ICCCC 2008).
[7] RacerPro Reasoner, http://www.racer-systems.com/
[8] World Wide Web Consortium. OWL Web Ontology Language Reference. W3C Recommendation, 2004

Ovidius University of Constanta, 124 Mamaia Blvd., 900527 Constanţa, Romania
E-mail address: cbogdan@univ-ovidius.ro