Shallow Natural Language Technology and Text Mining

Günter Neumann
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany

Sven Schmeier
XtraMind Technologies,
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany

Abstract

At the Language Technology Lab of DFKI we are developing advanced robust and efficient methods and components for free NL text processing which are suitable for data-intensive applications like text mining, information extraction or intelligent search engines. In this paper we will present a short overview of some of the core components, and how they have been used together with well-known Machine Learning tools as part of two application projects in the area of text mining, especially text classification.

1 Introduction

Text Mining (TM) is concerned with the task of extracting relevant information from natural language (NL) text documents and to search for interesting relationships between the extracted entities (i.e., structured data objects). A challenging feature of TM systems is that the information is only implicitly encoded in an unstructured way from the perspective of a computational system. Thus, a major first step in every TM system is to map the unstructured NL text to a structured internal representation (basically a set of data objects), which is then processed by the mining algorithms. It seems obvious that the more structure one can extract from the NL texts the better the mining algorithm might perform. In principle, it would be possible to use an exhaustive and deep generic text understanding system, which would aim to accommodate the full complexities of a language and to make sense of the entire text. However, even if it would be possible to formalize and represent the complete lexicon and grammar of a natural language, the system would still need a very high degree of robustness and efficiency. It has been shown that realizing such a system is at least today impossible. However, in order to fulfill the ever increasing demands for improved processing of real-world texts, NL researchers have started to relax the theoretical challenge to some more practical approaches which handle the requested robustness and efficiency. This has lead to so called shallow NL processing approaches, where certain generic language regularities which are known to cause complexity problems are either not handled, e.g., instead of computing all possible readings only an underspecified structure is computed or handled very pragmatical, e.g., by restricting the depth of recursion on the basis of a corpus analysis or by making us of heuristic rules, like ñlongest matching substringsò. This engineering view of language has lead to a renaissance and improvement of well-known efficient techniques, most notably finite state technology for parsing.

We are interested in exploring and investigating re-usable and domain-adaptive language technology by viewing NLP as a stepwise process of normalization. In this paper we describe the experience we obtained by combining shallow NL technology with machine learning tools in the area of text classification which was conducted as part of two application projects. The main questions we were (and still are) interested:

1. How deep do we have to analyse texts?
2. Which learning algorithm is the best and how many training examples do we need?
3. What can we expect for resulting accuracy?

The (obvious?) main answer we obtained was that it all depends on the data but our results gave us some heuristics and hints that might help in other application domains.

1.1 Project background

The first project **ICC** ï Innovation at the Call Center ï was founded by the Ministry of Economy and Finances of the Saarland and has been realized in close collaboration with **AOL**. Its motivation resulted from the observation that customer care in technical domains is increasingly based on e-mail communication, allowing for reproduction of approved solutions. For a Call Center agent, identifying the

customerôs problem is often time-consuming, as the problem space changes if new products are launched or existing regulations are modified. This task can partly be automated by a system suggesting relevant solutions for an incoming e-mail. Hence we developed an assistance system for automatic classification of emails to a set of predefined answering blocks, cf. [Busemann et al., 2000]. The second project **TIM** ï Telekom Information Management ï was funded by the German Telekom AG. The task was to produce a system for automatic press clipping, i.e. press releases should be automatically classified to several classes. Here, the main focus was on the systemôs precision, i.e. a text should be classified only if it is sure that it belongs to the class.

1.2 Shallow Natural Language Processing

In both projects, linguistically based preprocessing of text documents is performed by SMES, an information extraction core system for real world German text processing [Neumann et al., 1997, Neumann et al., 2000].

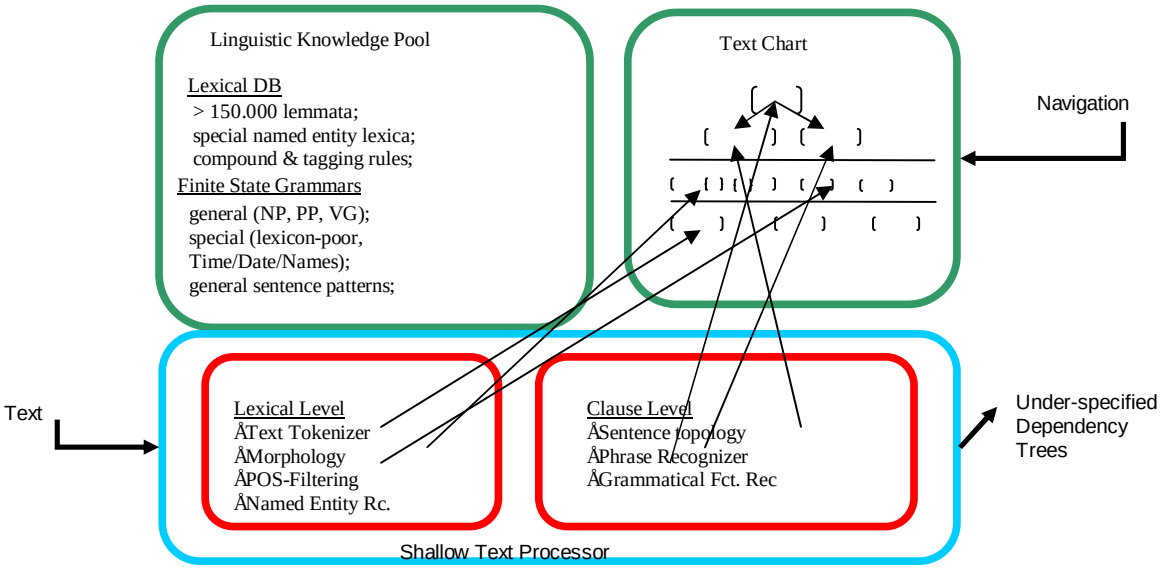

Fig1

It consists of two major components the Linguistic Knowledge Pool (LKP) and STP, the core shallow text processor of SMES (see Fig1). STP processes a NL-text through a chain of modules. We distinguish two primarily levels of processing, the *lexical level* and the *clause level*. Both are subdivided into several components. All lexical and grammatical components of SMES are realized by means of cascaded weighted finite state machines. The final result for a sentence computed by SMES is an *underspecified dependence structure*, where only upper bounds for attachment and scoping of modifiers are expressed.

Tokenization The tokenizer maps sequences of consecutive characters into larger units called tokens and identifies their types. Currently we use more than 50 token classes including generic classes for semantically ambiguous tokens (e.g., ñ10:15ò could be a time expression or volleyball result, hence we classify this token as number-dot compound) and complex classes like abbreviations or complex compounds (e.g., ñAT&T-Chiefò). It proved that such variety of token classes simplifies the processing of subsequent sub-modules significantly.

Morphology and Tagging Each token identified as a potential word form is analysed by the morphological analysis including on-line recognition of compounds (which is crucial since compounding is a very productive process of the German language) and hyphen coordination (e.g., in ñAn- und Verkaufò (*purchase and sale*) ñAn-ñ is resolved to ñAnkaufò (*purchase*)). Each token recognized as a valid word form is associated with the list of its possible readings, characterized by stem, inflection information (e.g., case, person, gender) and part of speech category. Since a high amount of German

word forms is ambiguous efficient disambiguation strategies are needed. Apart from manually constructed rules (taking into account of German specific spelling rules), we also used rules automatically determined by Brill's tagger, [Brill, 95].

Named entity finder Named entities (NE) such as organizations, persons, locations and time expressions are dynamically identified using finite-state grammars. Since some NEs (e.g. company names) may appear in the text either with or without a designator, we use a dynamic lexicon to store recognized NEs without their designators (e.g., ñBraun AGò vs. ñBraunò) in order to identify subsequent occurrences correctly. In the *whiteboard* project (see sec. 5) we have recently developed an unsupervised learning method for NE recognition based on Maximum Entropy Modeling which will be used to automatically extend the existing NE coverage of SMES.

Parsing In most of the well-known shallow text processing systems cascaded chunk parsers are used, which perform clause recognition after fragment recognition following a bottom-up style. We have also developed a similar bottom-up strategy for the processing of German texts. However, the main problem we experienced using the bottom-up strategy was insufficient robustness: because the parser depends on the lower phrasal recognizers, its performance is heavily influenced by their respective performance. As a consequence, the parser frequently wasn't able to process structurally simple sentences, because they contained, for example, highly complex nominal phrases.
For that reason we developed a novel top-down/bottom-up chunk parser, which consists of three major subcomponents.
During the first step of the parser a cascade of finite-state grammars are applied to the stream of lexical tokens and named entities in order to determine the *topological structure* of the sentence according to the linguistic field theory. A sentence is segmented into several parts: the front field, the left verb part, middle field, right verb part, and rest field. Subclauses can also be expressed in that way such that the left verb part is either empty or occupied by a relative pronoun or a subjunction element, and the complete verb group is placed in the right verb part. Note that each separated field can be arbitrarily complex with very few restrictions on the ordering of the phrases inside a field. After the *phrase recognizer* has expanded the corresponding phrasal strings, a further analysis step is done by the *grammatical function recognizer* which identifies possible *arguments* on the basis of the lexical subcategorization information available for the local head. The final output of the clause level for a sentence is thus an *underspecified dependence structure*. This is a flat dependence-based structure of a sentence, where only upper bounds for attachment and scoping of modifiers are expressed.

SMES has a huge lexical database with more than 150.000 stem entries, more than 12,000 sub-categorization frames as well as basic lexicons for proper names. During several experiments we observed that SMES has a very good performance and coverage (e.g., 97,9 % tagging accuracy, a precision of 95.77 % and a recall of 85 % for the NE recognizer, and about 87% F-measure for the chunk parser), cf. [Neumann et al., 2000]. Complete processing of a text with about 4500 words takes about 1 second. Very important in the context of this paper is SMES's high degree of modularity: each component can be used in isolation. Thus, it is possible to run only a subset of the components, e.g. to perform term extraction by using only the specialized sub-grammars or/and the phrasal grammars.

1.2 Machine Learning Software

For the task of text classification, several Machine Learning tools, which stand for different learning paradigms, have been selected and evaluated in different settings of our domains: the decision tree learners ID3 [Quinlan, 1986] and MC4 which combines ID3 and pruning methods of C4.5 [Quinlan, 1992], the rule-based learner RIPPER [Cohen, 1995] and its boosted version, and the support vector machine learner SVM-Light [Joachims, 1998]. Recently, we conducted additional experiments for email classification with a ROCCHIO-like [Lewis *et al*, 1996] relevance feedback algorithm and a SVM implementation based on SMO [Platt, 1999] which promise even better results. Unfortunately up to now we did not perform tests using the same preconditions as with the other systems.

2 Data

The kind of data is essential for SMES and ML systems. In our scenarios the press releases have been of a much better quality than the emails. For instance, in emails punctuation marks are used very

loosely; we had to cope with a large amount of misspellings, and most emails lacked grammatical correctness, as can be seen in the following example:
ò Wie mache ich zum mein Programm total deinstalieren, und wieder neu instalierem, mit, wen Sie mir senden Version 4.0 ??????????????ò which roughly translates to: òHow do I make to mine program totally deinstal, and again new reinstall, with, who you send to me version 4.0 ??????????????ò.
In general, the decision on how deep linguistic preprocession can be useful depends on the data (because of the pure recall expected). If you go too deep you might not get any results, whereas if you decide to stay on the surface, you will probably get problems in detecting structural similarities in data. Concerning the ML algorithms, the number of categories and training examples available (and their distribution among the categories) and the length of the texts are important parameters. In ICC there are 2350 training examples and 44 categories. The emails contain 60 words on average. In TIM there are 824 training examples and 6 categories. The press releases contain 578 words on average. This different information had important effects on our results. A further important aspect concerns the noisiness of data. It turned out that the data of **ICC** was noisier than that of **TIM** in the following sense:

- in some emails several problems are mentioned, i.e. several categories are appropriate
- the category system is ambiguous by offering several classes for the same problems (highlighting several sub-themes)
- the example corpus has been òcreatedò by the clerks in the call center and has not been supervised by some expert(s).

We studied the influence of non-linguistic and linguistic preparation on data that is fed to the ML programs by selecting a subset of SMESôs components. We tried simple letter trigrams, morphological analysis of nouns, verbs and adjectives, and shallow parsing methods that extract domain specific information. The main issue of the ñdeeperò linguistic preprocessing is the combination of domain knowledge with pure statistical methods of the ML part. Furthermore we experimented with several different feature weighting measures like binary, word counts, or several different relevance measures like tf-idf or Chi-Square. In general tf-idf measurements produced the best result except for SVM-Light where we used pure word counts.

3 Results

The kind of measurement is ñaccuracyò which means in average X percent of incoming new texts will be classified into the correct category. All measurements were done using 10 fold cross validation.

3.1 ICC Experiments

We made several tests using different data preprocessing for the ML algorithms. We started our test series with letter trigrams. The second preprocessing has been morphological analysis. Here the Learner was fed with unknown words and the stems of nouns, verbs, adjectives and adverbs. The last kind of data preprocessing consisted of shallow text processing. We found that main information of emails in a Call Center is to be found in sentences containing negation and special words as well as in questions such as: ñWhy canôt I read my email attachments?ò or ñI canôt read my email attachments.ò. The results are:

	trigram	morph	shallow
ID3	28.45	46.82	44.55
MC4	29.29	47.45	45.67
RIPPER	47.12	56.11	56.92
Boosted R	52.78	60.37	60.76
SVM-Light	54.29	58.29	61.42

This test series show that morphological analysis of short German texts seems to be a better choice than simple trigramming. The decision whether to introduce additional knowledge by using shallow parsing should depend on the used ML algorithm.

3.2 TIM Experiments

In TIM we didnôt proceed to deeper SMES yet but we intend doing this in future research. The results were:

	trigram	morph
ID3	74.44	75.33
MC4	72.35	72.00
RIPPER	71.01	70.34
Boosted R	79.55	79.67
SVM-Light	81.25	81.67

Again SVM-Light outperformed all other learning algorithms. Furthermore, because SVM-Light provides the possibility to take only examples into account that cause high confidence values it can be tuned for high-precision classification. Morphological analysis does not significantly raise the classification accuracy. The reason is that the documents are much larger thus linguistic normalization of words is not that important. However, it might also be the case, that because of the low number of different classes of the TIM corpus (6 classes), the ML learners are not ñforced to understandò the texts in certain detail, because ñmeaning differencesò are sufficiently represented in character sequences.

3 Ongoing and future work

For complex tasks like the discovery of domain-specific relations from real-world NL texts, more complex NL components have to be investigated, e.g., accurate analysis of grammatical functions, deep case or anaphora resolution. Hence, the necessary amount of deep processing seems to increase with the complexity of the tasks. In the DFKI project *whiteboard*[1], methods and technologies for the integration of shallow and deep NLP are explored. One main strategy we are following could be paraphrased as ñdemand-driven control ò: only call deep NL components if a more detailed and accurate analysis is requested for the relevant data objects computed by the shallow components. The other main strategy we are following is the automatic domain-specific extraction and extension of general linguistic resources using hybrid machine learning methods, e.g., in order to bootstrap domain-specific lexical and grammatical knowledge, cf. [Neumann, 2001]. In both cases we hope to be able to combine the advantage of deep (precise and exhaustive) and shallow (robust and efficient) natural language methods into one overall model.

References

[Brill, 95] Eric Brill. Transformation-based error-driven learning and natural language processing: *A case study in part-of-speech tagging*. Computational Linguistics, 21(4):543-565, December 1995.

[Busemann *et al*, 2000] Stephan Busemann, Sven Schmeier and Roman G. Arens: *Message Classification in the Call Center*. In Proceedings of 6th ANLP, Seattle, 2000.

[Cohen, 1995] William W. Cohen. *Fast Effective Rule Induction* In ML95, 1995

[Joachims, 1998] T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* Proceedings of the European Conference on Machine Learning, Springer, 1998.

[Lewis *et al*, 1996] David D. Lewis, Robert Shapire, James P. Callan and Ron Papka. *Training Algorithms for Linear Text Classifiers.* SIGIR, 1996

[Neumann *et al*, 1997] G. Neumann, R. Backofen, J. Baur, M. Becker, C. Braun. *An Information Extraction Core System for Real World German Text Processing*. In Proceedings of 5th ANLP, Washington, March, 1997.

[Neumann *et al*, 2000] G. Neumann, C. Braun and J. Piskorski. *A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts.* In Proceedings of 6th ANLP, Seattle, 2000.

[1] The whiteboard project is funded by the BMBF from 2000 to 2002. Project principal investigator is Prof. H. Uszkoreit, project leader is Dr. G. Neumann. For more information see http://www.dfki.de/pas/f2w.cgi?ltp/whiteboard-e

[Neumann, 2001] G. Neumann. *Data-driven Approaches to Head-driven Phrase Structure Grammar*. In Rens Bod, Remko Scha and Khahil Sima'an (eds.): Introduction to Data-oriented Parsing. CSLI publications, Stanford, to appear.

[Quinlan, 1986] J.R. Quinlan. *Induction of Decision Trees*. Machine Learning, 1:81-106, 1986.

[Quinlan, 1992] J.R.Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, Los Alto, California , 1992.

[Platt, 1999] J.Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. In Advances in Kernel Methods - Support Vector Learning, pages 255-268. MIT Press, Cambridge, MA, 1999.