

Language models as feature extractors for term detection using SVM

Jody Foo
jodfo@ida.liu.se

May 20, 2009

Abstract

This paper presents a novel use of language models as feature sources for term detection using Support Vector Machines (SVMs). Two separate language models were built using the SRILM toolkit. One using the written texts of the British National Corpus, the second using patent texts from the C04B subclass concerning Lime, Magnesia and Slag. Terms from the C04B (2552 terms) subclass were mixed with 10 000 randomly sampled words from the Webster's Second International dictionary. The terms and dictionary words were scored using the two language models. The scores were then used to create a SVM for term detection. The test set of 781 was classified into terms and non-terms with a accuracy of 96.5%

1 Background

The field of computational terminology is a fairly new research area. Early research was mostly done by classical terminologists wanting to use new database technologies to improve the current methods.

Recent research however, also deals with computational methods for term extraction, terminology standardization and updating existing terminologies to name a few tasks. As terminologies have similarities with ontologies, some methods and issues are shared as well.

Jacquemin [3] puts term extraction in the field of Term-oriented NLP. This area is divided into four areas as seen in The four main subdomains of term-based NLP (Table 1)

Table 1: The four main subdomains of term-based NLP

	Prior terminological data	No prior terminological data
Term discovery	Term enrichment	Term acquisition
Term recognition	Controlled indexing	Free indexing

1.1 Term extraction

Term extraction is the process of extracting terms from texts. The exact definition of what a term is, however depends of the context. A classical approach to terminology is that terms denote concepts, and without a concept, there are no terms.

A more computational approach to terms has been to see them as *domain specific vocabulary*[3]. There are of course connections between these two views, and finding *domain specific vocabulary* might be a starting point in creating or standardizing terminology in the classical sense. For the remaining of this paper I will use *term* in its more computationally oriented sense.

The problem area of acquiring terms has been called *term extraction* in current research. Automatic Term Recognition (ATR) [1] is a similar, if not an overlapping area of research.

1.2 The process of term extraction

The most common approach to monolingual term extraction is to first examine the set of documents and produce term candidates. These term candidates are then validated in some way and the output is a set of validated terms.

One task in the problem area of term extraction is to classify an extracted term candidate as either a term or as general language in the context of the current domain. Manual term classification requires domain knowledge and the task takes time and is quite repetitive and boring for humans. In many cases, the number of extracted terms is quite large which means that considerable resources have to be used to create a manually validated terminology.

Automating or improving the classification process using computational methods can be done on several levels. A good ranking metric for example would increase the term yield per man hour spent on manual validation. Another way of improving the classification process is to reduce the amount of term candidates by removing term candidates that belong to the general vocabulary [6].

1.3 Language models

Language models are used in many areas of research to estimate how likely a certain input is for a given language. In speech recognition language models are used to choose between different interpretations of the audio signal whereas in statistical machine translation, language models are used to decide which translation is more probable in the target language.

A statistical language model uses observed word sequences, *n-grams* to model what is likely and less likely to be expressed in the observed language. In most cases, word sequences of $n=2, 3$ and 4 are used [5].

The basic language model tries to estimate the sequence probability $P(w_n|w_1...w_{n-1})$ and there has been different proposals on how to produce such an estimation. I will however not discuss the particulars of these approaches in this paper as a better overview can be found in other places such as [5].

1.4 Using language models to produce termhood features

Tomokiyo et al. [8] describes a method of using multiple language models and pointwise KL-divergence to extract keyphrases. In [8] they introduce the concepts of a *foreground* and a *background corpora*, as well as a *foreground* and a *background language model*. The

probability outputs from the language models are compared using pointwise KL-divergence to assess the importance of the keyphrase.

The hypothesis behind the experiments presented in this paper is that language models can be used to guide us in deciding whether or not a word or a term candidate belongs to *domain specific vocabulary*, the foreground corpus/language model or the *general vocabulary*, the background corpus/language model. However, since the task is concerned with term candidate classification, rather than importance scoring, I have chosen to use a classifier instead of a comparative ranking metric.

There seems to have been little or no research on using language model output as classification features, making this a novel method.

2 Method

The approach proposed is to use a domain specific language model and a language model created from a balanced corpus to score words or term candidates. These scores can then be used as features in a classification algorithm.

2.1 Dataset

This section describes the data used in the following experiment.

2.2 Manually verified terminology

The terms used in the experiment was a set of 2552 manually verified English patent terms (using domain experts), which were extracted from patent documents in the C04B subclass (Lime, Magnesia and Slag).

2.3 General language dictionary

Webster's Second International, 234,936, published 1934 (available in /usr/share/dict/ on Mac OS X 10.5). 10 000 random entries were selected. All entries from the dictionary were considered to be non-terms.

2.4 Language Models

Two 3-gram language models were used. The general language model was created using the entire text section of the British National Corpus¹, and the domain specific language model was created using the documents from which the validated terms were extracted from, i.e. patent documents belonging to the C04B subclass.

The general language model contained 39802 unigrams, 396908 bigrams and 284664 trigrams. The domain specific language model contained 559075 unigrams, 10837636 bigrams and 8234861 trigrams.

Both the BNC texts and the patent texts were processed in the same fashion, tokenized using version 3 of the Moses tokenizer [4], lowercased and build using the SRI Language Modeling Toolkit [7].

¹<http://www.natcorp.ox.ac.uk/>

2.5 Experiments

There are several variables that can be analyzed regarding the proposed method, e.g. how large must the domain specific and general language corpora be to be of use? Does the ratio between the domain specific and general language corpora have an effect on the result? Also, what scope should be covered by the domain specific corpora, a narrow area or a wider area?

This paper presents one experiment which is meant to serve as a pilot study, or proof of concept. The experiment examines whether or not a SVM is able to distinguish between general vocabulary and terms using features acquired from language model scores.

The experiment uses two language models described previously, one created using the British National Corpus, and one created using patent documents from the C04B patent subclass.

Manually validated patent terms from were intermixed with dictionary entries and assigned features acquired via language model scoring. The data was partitioned into a train set, a devtest set and a test set. Using the training set, a SVM classifier was trained and used to classify the test sets.

2.6 Features from language models

The patent terms and the general language dictionary entries were scored using the BNC100 and C04B language models. The scores used as features were the log probability (logprob) and the perplexity given using two different normalizations (ppl0 and ppl1).

2.7 Partitions

The following partitions sizes were used:

- Train: 10162
- Devtest: 1563
- Test: 781

2.8 Term detection using SVM

The SVM framework used in the experiment was the *e1071* package in R² which interfaces with the libsvm library [2]. Using the package the training data set was fed into a SVM using a radial kernel which was then tuned using a grid search to find the best cost (C) and gamma parameters.

3 Results

The SVM was able to classify the devtest data with an accuracy of 95.8% and the test data with an accuracy of 96.5%. The devtest set results (Table 2) and test set results (Table 3) show the results from the devtest and test set predictions using the tuned SVM.

²<http://www.r-project.org/>

Table 2: devtest set results

pred	0	1
0	1219	26
1	39	279

Table 3: test set results

pred	0	1
0	601	13
1	14	153

4 Conclusion and future research

The approach of using language models to provide features for SVM classification of terms and non-terms was successful in this pilot study. However additional studies must be made to confirm the results.

4.1 Future research

Various elaborations and variations can be made to the experimental design such as

- varying the size of the corpora used to create the language models
- varying the relationship between the documents used to create the domain specific language model and the documents from which the terms were extracted
- using words sampled from the domain specific documents rather than dictionary entries to add noise to the classification set

Also the language model based features must be compared with existing features used to describe termness such as tf-idf, C-value and NC-value [9].

References

- [1] ANANIADOU, S. A methodology for automatic term recognition. *Proceedings of the 15th conference on Computational Linguistics* (Jan 1994). 2
- [2] DIMITRIADOU, E., HORNIK, K., LEISCH, F., MEYER, D., AND WEINGESSEL, A. Package 'e1071'. 4
- [3] JACQUEMIN, C., AND BOURIGAULT, D. Term extraction and automatic indexing. *Handbook of computational linguistics* (Jan 2003). 1, 2
- [4] KOEHN, P., HOANG, H., BIRCH, A., AND CALLISON-BURCH, C. Moses: Open source toolkit for statistical machine translation. *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS* (Jan 2007). 3
- [5] MANNING, C. D., AND SCHÜTZE, H. Foundations of statistical natural language processing. 2
- [6] MERKEL, M., AND FOO, J. Terminology extraction and term ranking for standardizing term banks. *Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007. University of Tartu, Tartu, 2007.* (May 2007), pp. 349–354. 2
- [7] STOLCKE, A. Srlm-an extensible language modeling toolkit. *Seventh International Conference on Spoken Language Processing* (2002). 3
- [8] TOMOKIYO, T., AND HURST, M. A language model approach to keyphrase extraction. *Proceedings of the ACL Workshop on Multiword Expressions* (2003), 33–40. 2
- [9] ZHANG, Z., JOSÉIRIA, C., AND CIRAVEGNA, F. A comparative evaluation of term recognition algorithms. *dcs.shef.ac.uk* (2008). 5