Automatic recognition of domain-specific terms: an experimental evaluation

© Denis Fedorenko Nikita Astrakhantsev Denis Turdakov

Institute for System Programming of Russian Academy of Sciences
fedorenko@ispras.ru, astrakhantsev@ispras.ru, turdakov@ispras.ru

Abstract

This paper presents an experimental evaluation of the state-of-the-art approaches for automatic term recognition based on multiple features: machine learning method and voting algorithm. We show that in most cases machine learning approach obtains the best results and needs little data for training; we also find the best subsets of all popular features.

1 Introduction

Automatic term recognition (ATR) is an actual problem of text processing. The task is to recognize and extract terminological units from different domain-specific text collections. Resulting terms can be useful in more complex tasks such as semantic search, question-answering, ontology construction, word sense induction, etc.

There have been a lot of studies of ATR. Most of them split the task into three common steps:

1. **Extracting term candidates.** At this step special algorithm extracts words and word sequences admissible to be terms. In most cases researches use predefined or generated part-of-speech patterns to filter out word sequences that do not match such the patterns. The rest of word sequences becomes term candidates.

2. **Extracting features of term candidates.** Feature is a measurable characteristic of a candidate that is used to recognize terms. There are a lot of statistical and linguistic features that can be useful for term recognition.

3. **Extracting final terms from candidates.** This step varies depending upon the way in which researches use features to recognize terms. In some studies authors filter out non-terms by comparing feature values with thresholds: if feature values lies in specific ranges, then candidate is considered to be a term. Others try to rank candidates and expect the top-N ones to be terms. At last, few studies apply supervised machine learning methods in order to combine features effectively.

Proceedings of the Ninth Spring Researcher's Colloquium on Database and Information Systems, Kazan, Russia, 2013

There are several studies comparing different approaches for ATR. In [18] authors compare different single statistical features by their effectiveness for term candidates ranking. In [24] the same comparison is extended by voting algorithm that combines multiple features. Studies [17], [15] compare supervised machine learning method with the approach based on single feature again.

In turn, the present study experimentally evaluates the ranking methods combining multiple features: supervised machine learning approach and voting algorithm. We pay most of the attention to the supervised method in order to explore its applicability to ATR.

The purposes of the study are the following:

- To compare results of machine learning approach and voting algorithm;
- To compare different machine learning algorithms applied to ATR;
- To explore how much training data is needed to rank terms;
- To find the most valuable features for the methods;

This study is organized as follows. At the beginning we describe the approaches more detailed. Section 3 is devoted to the performed experiments: firstly, we describe evaluation methodology, then report the obtained results, and, finally, discuss them. In Section 4 we conclude the study and consider the further research.

2 Related work

In this section we describe some of the approaches to ATR. Most of them have the same extracting algorithm but consider different feature sets, so the final results depend only on the used features. We also briefly describe features used in the task. For more detailed survey of ATR see [10], [2].

2.1 Extracting term candidates overview

Strictly, all of the word sequences, or *n-grams*, occurring in text collections can be term candidates. But in most cases researchers consider only unigrams and bigrams [18]. Of course, only the little part of such the candidates are terms, because the candidates' list mainly consists of sequences like "a", "the", "some of", "so the", etc. Hence such the noise should be filtered out.

One of the first methods for such the filtering was described in [12]. The algorithm extracts term candidates by matching the text collection with predefined Part-of-Speech (PoS) patterns, such as:

- Noun
- Adjective Noun
- Adjective Noun Noun

As was reported in [12], such the patterns cut off much of the noise (word sequences that are not terms) but retain real terms, because in most cases terms are noun phrases [5]. Filtering of term candidates that do not satisfy some of the morphological properties of word sequences is known as *lingustic step* of ATR.

In work [17] the authors do not use predefined patterns appealing to the fact that PoS tagger can be not precise enough on some texts; they instead generate patterns for each text collection. In study [7] no linguistic step is used: the algorithm considers all n-grams from text collection.

2.2 Features overview

Having a lot of term candidates, it is necessary to recognize domain specific ones among them. It can be done by using the statistical features computed on the basis of the text collection or some another resource, for example general corpus [12], domain ontology [23] or Web [6]. This part of ATR algorithm is known as *statistical step*.

Term Frequency is a number of occurrences of the word sequence in the text collection. This feature is based on the assumption that if the word sequence is specific for some domain, then it often occurs in such domain texts. In some studies frequency is also used as an initial filter of term candidates [3]: if a candidate has a very low frequency, then it is filtered out. It helps to reduce much of the noise and improves precision of the results.

*TF*IDF* has high values for terms that often occur only in few documents: TF is a term frequency and IDF is an inversed number of documents, where the term occurs:

$$TF * IDF(t) = TF(t) \cdot \log \frac{|Docs|}{|\{Doc : t \in Doc\}|} \quad (1)$$

To find domain-specific terms that are distributed on the whole text collection, in [12] IDF is considered as an inversed number of documents in *reference corpus*, where the term occurs. Reference corpus is a some general, i.e. not specific, text collection.

The described features shows how the word sequence is related to the text collection, or *termhood* of a candidate. There is another class of features that show inner strength of words cohesion, or *unithood* [10]. One of the first features of this class is T-test.

T-test [12] is a statistical test that was initialy designed for bigrams and checks the hypothesis of independence of words constituting a term:

$$T\text{-}stat(t) = \frac{\frac{TF(t)}{N} - p}{\sqrt{\frac{p(1-p)}{N}}} \quad (2)$$

where p - hypothesis of independence, N - a number of bigrams in the corpus.

The assumption of this feature is that the text is a Bernoulli process, where meeting of bigram t is a "success", while meeting of other bigrams is a "failure".

Hypothesis of independence is usually expressed as follows: $p = P(w_1 w_2) = P(w_1) \cdot P(w_2)$, where $P(w_1)$ - a probability to encounter the first word of the bigram, $P(w_2)$ - a probability to encounter the second one. This expression can be assessed by replacing the probabilities of words to their normalized frequencies within a text: $p = \frac{TF(w_1)}{N} \cdot \frac{TF(w_2)}{N}$, where N - an overall number of words in the text.

If words are independently distributed in text collection, then they do not form persistent *collocation*. It is assumed that any domain-specific term is a collocation, while not any collocation is a specific term. So considering features like T-test, we can increase the confidence in that candidate is a collocation, but not necessarily specific term.

There are much more features that are used in ATR.

C-Value [8] has higher values for candidates that are not parts of other word sequences:

$$C\text{-}Value(t) = \log_2 |t| \cdot TF(t) - \\ - \frac{1}{|\{seq : t \in seq|\}} \sum_{t \in seq} TF(seq) \quad (3)$$

Domain Consensus [14] recognizes terms that are uniformly distributed on the whole dataset:

$$DC(t) = - \sum_{d \in Docs} \frac{TF_d(t)}{TF(t)} \log_2 \frac{TF_d(t)}{TF(t)} \quad (4)$$

Domain Relevance [20] compares frequencies of the term in two datasets - target and general:

$$DR(t) = \frac{TF_{target}(t)}{TF_{target}(t) + TF_{reference}(t)} \quad (5)$$

Lexical Cohesion [16] is the unithood feature that compares frequency of term and frequency of words from which it consists:

$$LC(t) = \frac{|t| \cdot TF(t) \cdot \log_{10} TF(t)}{\sum_{w \in t} TF(w)} \quad (6)$$

Loglikelihood [12] is the analogue of T-test but without assumption about how words in a text are distributed:

$$LL(t) = \log \frac{b(c_{12}; c_1, p) b(c_2 - c_{12}; N - c_1, p)}{b(c_{12}; c_1, p_1) b(c_2 - c_{12}; N - c_1, p_2)} \quad (7)$$

where c_{12} - a frequency of bigram t, c_1 - a frequency of the bigram's the first word, c_2 - a frequency of the second one, $p = \frac{c_2}{N}$, $p_1 = \frac{c_{12}}{c_1}$, $p_2 = \frac{c_2 - c_{12}}{N - c_1}$, $b(\cdot; \cdot, \cdot)$ - binomial distribution.

Relevance [19] is the more sophisticated analogue of Domain Relevance:

$$R(t) = 1 - \frac{1}{\log_2(2 + \frac{TF_{target}(t) \cdot DF_{target}(t)}{TF_{reference}(t)})} \quad (8)$$

Weirdness [1] also compares frequencies in different collections but also takes into account sizes of such the collections:

$$W(t) = \frac{TF_{target}(t) \cdot |Corpus_{reference}|}{TF_{reference}(t) \cdot |Corpus_{target}|} \quad (9)$$

The described feature list includes termhood, unithood and *hybrid* features. The termhood features are Domain Consensus, Domain Relevance, Relevance, and Weirdness. The unithood features are Lexical Cohesion and Loglikelihood. The hybrid feature, or feature that shows both termhood and unithood, is C-Value.

A lot of works still concentrate on feature engineering, trying to find more informative features. Nevertheless, recent trend is to combine all these features effectively.

2.3 Recognizing terms overview

Having feature values, final results can be produced. The studies [8], [12], [1] use ranking algorithm to provide the most probable terms, but this algorithm considers only one feature. The studies [20], [16] describe the simplest way of how multiple features can be considered: all values are simply reduced in a one weighted average value that then is used during ranking.

In work [21] authors introduce special rules based on thresholds for feature values. An example of such a rule is the following:

$$Rule_i(t) = F_i(t) > a \text{ and } F_i(t) < b \quad (10)$$

where F_i is a i-th feature; a, b are thresholds for feature values.

Note that the thresholds are selected manually or computed from the marked-up corpora, so this method can not be considered as purely automatic and unsupervised.

Effective way of combining multiple features was introduced in [24]. It combines the features in a voting manner using the following formula:

$$V(t) = \sum_{i}^{n} \frac{1}{rank(F_i(t))} \quad (11)$$

where n is a number of considered features, $rank(F_i(t))$ is a rank of the term t among values of other terms considering feature F_i.

In addition, study [24] shows that the described voting method in general outperforms most of the methods that consider only one feature or reduce them in a weighted average value. Another important advantage of the voting algorithm is that it does not require normalization of feature values.

There are several studies that apply supervised methods for term recognition. In [17] authors apply Ada Boost meta-classifier, while in [7] Ripper system is used. The study [22] describes hybrid approach including both unsupervised and supervised methods.

Dataset	Algorithm	AvP
GENIA	Random Forest	0.54
GENIA	Logistic Regression	**0.55**
GENIA	Voting	0.53
Bio1	Random Forest	0.35
Bio1	Logistic Regression	**0.40**
Bio1	Voting	0.23

Table 1: Results of cross-validation without frequency filter

Dataset	Algorithm	AvP
GENIA	Random Forest	0.66
GENIA	Logistic Regression	**0.70**
GENIA	Voting	0.65
Bio1	Random Forest	0.52
Bio1	Logistic Regression	**0.58**
Bio1	Voting	0.31

Table 2: Results of cross-validation with frequency filter

3 Evaluation

For our experiments we implemented two approaches for ATR. We used voting algorithm as the first one, while in supervised case we trained two classifiers: Random Forest and Logistic Regression from WEKA library [1]. These classifiers were chosen because of their effectiveness and good generalization ability of the resulting model. Furthermore, these classifiers are able to produce *classification confidence* - a numeric score that can be used to rank an example in overall test set. It is an important property of the selected algorithms that allows to compare their results with results produced by other ranking methods.

3.1 Evaluation methology

The quality of the algorithms is usually assessed by two common metrics: precision and recall [11]. *Precision* is the fraction of retrieved instances that are relevant:

$$P = \frac{|\text{correct returned results}|}{|\text{all returned results}|} \quad (12)$$

Recall is the fraction of relevant instances that are retrieved:

$$R = \frac{|\text{correct returned results}|}{|\text{all correct results}|} \quad (13)$$

In addition to *precision* and *recall* scores, Average Precision (AvP) [12] is commonly used [24] to assess ranked results. It defines as:

$$\sum_{i=1}^{N} P(i)\Delta R(i) \quad (14)$$

where $P(i)$ is the precision of top-i results, $\Delta R(i)$ change in recall from top-(i-1) to top-i results.

Obviously, this score tends to be higher for algorithms that print out correct terms on top positions of the result.

In our experiments we considered only the AvP score, while precision and recall are omitted. For voting algorithm it is no simple way to compute recall, because it is

[1] Official website of the project: http://www.cs.waikato.ac.nz/ml/weka/

not obvious what number of top results should be considered as correct terms. Also in a general case the overall number of terms in dataset is unknown.

3.2 Features

For our experiments we implemented the following features: C-Value, Domain Consensus, Domain Relevance, Frequency, Lexical Cohesion, Loglikelihood, Relevance, TF*IDF, Weirdness and Words Count. Words Count is the simple feature that shows a number of words in a word sequence. This feature may be useful for the classifier since values of other features may have different meanings for single- and multi-word terms [2].

Most of these features are capable to recognize both single- and multi-word terms, except T-test and Loglikelihood that are designed to recognize only two-word terms (bigrams). We generalize them to the case of n-grams according to the study [4].

Some of the features consider information from the collection of general-domain texts (reference corpus), in our case these features are Domain Relevance, Relevance, Weirdness. For this purpose we use statistics from Corpus of Contemporary American English [2].

For extracting term candidates we implemented simple approach based on predefined part-of-speech patterns. For simplicity, we extracted only unigrams, bigrams and trigrams by using patterns such as:

1. Noun
2. Noun Noun
3. Adjective Noun
4. Noun Noun Noun
5. Adjective Noun Noun
6. Noun Adjective Noun

3.3 Datasets

Evaluation of the approaches was performed on two datasets of medical and biological domains consisting of short English texts with marked-up specific terms:

Corpus	Documents	Words	Terms
GENIA	2000	400000	35000
Bio1	100	20000	1200

The last one (Bio1) has common texts with the first (GENIA), so we filtered out the texts that occur in both the corpora. We left GENIA without any modifications, while 20 texts were removed from Bio1 as common texts of the corpora.

3.4 Experimental results

3.4.1 Machine learning method versus Voting algorithm

We considered two test scenarios in order to compare quality of the implemented algorithms. For each scenario we performed two kinds of tests: with and without filtering of rare term candidates.

[2]Statistics available at www.ngrams.info

Trainset	Testset	Algorithm	AvP
GENIA	Bio1	Random Forest	0.30
GENIA	Bio1	Logistic Regression	**0.35**
–	Bio1	Voting	0.25
Bio1	GENIA	Random Forest	0.44
Bio1	GENIA	Logistic Regression	0.42
–	GENIA	Voting	**0.55**

Table 3: Results of evaluation on separated train and test sets without frequency filter

Trainset	Testset	Algorithm	AvP
GENIA	Bio1	Random Forest	0.34
GENIA	Bio1	Logistic Regression	**0.48**
–	Bio1	Voting	0.31
Bio1	GENIA	Random Forest	0.60
Bio1	GENIA	Logistic Regression	0.62
–	GENIA	Voting	**0.65**

Table 4: Results of evaluation on separated train and test sets with frequency filter

In the following tests the whole feature set was considered and the overall ranked result was assessed.

Cross-validation

We performed 4-fold cross-validation of the algorithms on both the corpora. We extracted term candidates from the whole dataset and divided them on train and test sets. In other words, we considered the case when having some marked-up examples (train set) we should recognize terms in the rest of data (test set) extracted from the same corpus. So in case of voting algorithm the training set was simply omitted.

The results of cross-validation are shown in the Tables 1, 2. The Table 2 presents results of cross-validation on term candidates that appears at least two times in the corpus.

As we can see, in both the cases machine learning approach outperformed voting algorithm. Moreover, in the case without rare terms a difference of scores is higher. It can be explained by the following: feature values of rare terms (especially Frequency, Domain Consensus) are useless for the classification and add a noise to the model. When such the terms are omitted, the model becomes more clear.

Also in most cases Logistic Regression algorithm outperformed Random Forest, so in most of further tests we used only the best one.

Separate train and test datasets

Having two datasets of the same field, the idea is to check how the model trained on the one can predict the data from the other. For this purpose we used GENIA as a training set and Bio1 as a test one, then visa versa.

The results are shown in the Tables 3, 4. In the case when Bio1 was used as a training set, voting algorithm outperformed trained classifier. It could happen due to the fact that the training data from Bio1 does not fully reflect properties of terms in GENIA.

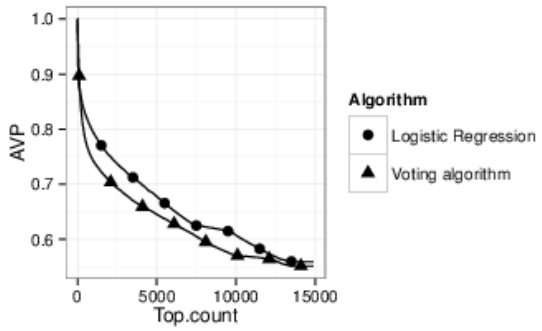

Figure 1: Dependency of AvP from top results given by cross-validation

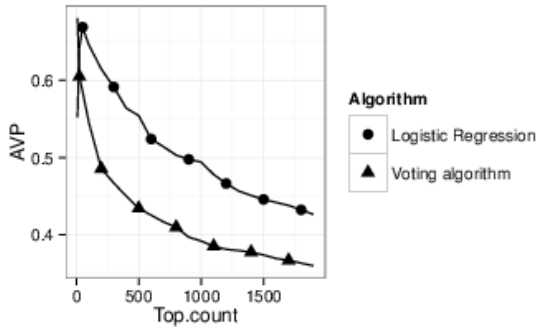

Figure 2: Dependency of AvP from top results on separated train and test sets

3.4.2 Dependency of average precision from number of top results

In previous tests we considered overall results produced by the algorithms. Descending from the top to the bottom of the ranked list, AvP score can significantly change, so one algorithm can outperform another one on top-100 results but lose on top-1000. In order to explore this dependency, we measured AvP for different slices of the top results.

The Figure 1 shows the dependency of AvP from number of top results given by 4-fold cross-validation.

We also considered a scenario when GENIA was used for training and Bio1 for testing. The results are presented on the Figure 2.

3.4.3 Dependency of classifier performance from training set size

In order to explore dependency between the amount of data used for training and average precision, we considered three test scenarios.

At first, we trained the classifiers on GENIA dataset and tested it on Bio1. At each step the amount of training data was being decreased, while the test data remained without any modifications. The results of the test are presented on the Figure 3.

Next, we started with 10-fold cross-validation on GENIA and at each step decreased the number of folds used for training of Logistic Regression and did not change the number of folds used for testing. The results are shown on the Figures 4–8.

The last test is the same as the previous one, except

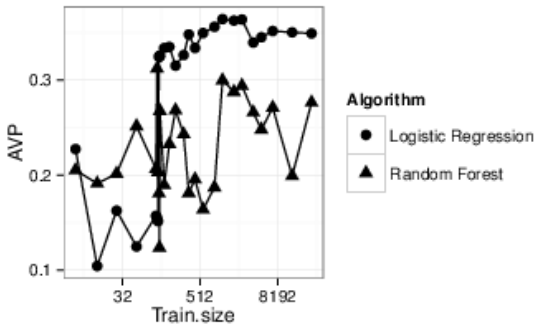

Figure 3: Dependency of AvP from train set size on separated train and test sets

that the number of test folds was being increased at each step. So we started with nine folds used for training and one fold used for the test. At the next step we moved one fold from training set to the test set and evaluated again. The results are presented on the Figures 9–13. The interesting observation is that higher values of AvP correspond to the bigger sizes of the test set. It could happen because with increasing of the test set the number of high-confident terms is also growing: such the terms take most of the top positions of the list and improve AvP. In case of GENIA and Bio1 the top of the list mainly consists from the highly domain-specific terms that take high values for the features like Domain Relevance, Relevance, Weirdness: such the terms occur in the corpora frequently enough.

As we can see, in all of the cases the gain of AvP stopped quickly. So, in case of GENIA, it is enough to train on 10% of candidates to rank the rest 90% with the same performance. It could happen because of the relatively small number of features are used and their specificity: most of them designed to have high magnitude for terms and low for non-terms. So, the data can be easily separated by the classifier having few training examples.

3.5 Feature selection

Feature selection (FS) is the process of finding the most relevant features for the task. Having a lot of different features, the goal is to exclude redundant and irrelevant ones from the feature set. *Redundant* features provide no useful information as compared with the current feature set, while *irrelevant* features do not provide information in any context.

There are different algorithms of FS. Some of them rank separate features by relevance to the task, while others search subsets of features that get the best model for the predictor [9]. Also the algorithms differ by their complexity. Because of big amount of features used in some tasks, it is not possible to do exhaustive search, so features are selected by greedy algorithms [13].

In our task we concentrated on searching the subsets of features that get the best results for the task. For such purpose we ran quality tests for all possible feature subsets, or, in other words, performed the exhaustive search. Having 10 features, we check $2^{10} - 1$ different combinations of them. In case of the machine learning method, we used 9 folds for test and one fold for train. The reason of such the configuration is that the classifier needs little

Top count	All features	The best features
100	0.9256	0.9915
1000	0.8138	0.8761
5000	0.7128	0.7885
10000	0.667	0.7380
20000	0.6174	0.6804

Table 5: Results of FS for voting algorithm

Top count	All features	The best features
100	0.8997	0.9856
1000	0.8414	0.8757
5000	0.7694	0.7875
10000	0.7309	0.7329
20000	0.6623	0.6714

Table 6: Results of FS for Logistic Regression

data for training to rank terms with the same performance (see the previous section). For voting algorithm, we simply ranked candidates and then assessed overall list. All of the tests were performed on GENIA corpus and only the Logistic Regression was used as the machine learning algorithm.

The AvP score was computed for different slices of the top terms: 100, 1000, 5000, 10000, and 20000. The same slices are used in [24]. The best results for the algorithms are presented in the Tables 5, 6. These tables shows that voting algorithm has better scores then machine learning method, but such the results are not fully comparable: FS for voting algorithm was performed on the whole dataset, while Logistic Regression was trained on 10% of term candidates. The average performance gain for voting algorithm is about 7%, while for machine learning it is only about 3%.

The best features for voting algorithm:

1. **Top-100**: Relevance, TF*IDF
2. **Top-1000**: Relevance, Weirdness, TF*IDF
3. **Top-5000**: Weirdness
4. **Top-10000**: Weirdness
5. **Top-20000**: CValue, Frequency, Domain Relevance, Weirdness

The best features for the machine learning approach:

1. **Top-100**: Words Count, Domain Consensus, Normalized Frequency, Domain Relevance, TF*IDF
2. **Top-1000**: Words Count, Domain Relevance, Weirdness, TF*IDF
3. **Top-5000**: Words Count, Frequency, Lexical Cohesion, Relevance, Weirdness
4. **Top-10000**: Words Count, CValue, Domain Consensus, Frequency, Weirdness, TF*IDF
5. **Top-20000**: Words Count, CValue, Domain Relevance, Weirdness, TF*IDF

As we can see, most of the subsets contain features based on a general domain. The reason can be that the target corpus has high specificity, so the most of terms do not occur in a general corpus.

The next observation is that in case of the machine learning algorithm, Words Count feature occurs in all of the subsets. This observation confirms an assumption that this feature is useful for algorithms that recognize both the single- and multi-word terms.

3.6 Discussion

Despite the fact that filtering of the candidates occurring only once in the corpus improves average precision of the methods, it is not always a good idea to exclude such the candidates. The reason is that a lot of specific terms can occur only once in a dataset: for example, in GENIA there are 50% of considered terms that occur only once. Of course, omitting such the terms extremely affects recall of the result. Thus such the cases should be considered for the ATR task.

One of the interesting observations is that the amount of training data is needed to rank terms without sufficient performance drop is extremely low. It leads to the idea of applying the bootstrapping approach for ATR:

1. Having few marked-up examples, train the classifier
2. Use the classifier to extract new terms
3. Use the most confident terms as initial data at step 1.
4. Iterate until all of confident terms will be extracted

This is a semi-supervised method, because only little marked-up data is needed to run the algorithm. Also the method can be transformed into fully unsupervised, if initial data will be extracted by some unsupervised approach (for example, by voting algorithm). The similar idea is implemented in study [22].

4 Conclusion and Future work

In this paper we have compared the performance of two approaches for ATR: machine learning method and voting algorithm. For this purpose we implemented the set of features that include linguistic, statistical, termhood and unithood feature types. All of the algorithms produced ranked list of terms that then was assessed by average precision score.

In most tests machine learning method outperforms voting algorithm. Moreover it was explored that for the supervised method it is enough to have few marked-up examples, about 10% in case of GENIA dataset, to rank terms with good performance. It leads to the idea of applying bootstrapping to ATR. Furthermore, initial data for bootstrapping can be obtained by voting algorithm because its top results are precise enough (see the Figure 1)

The best feature subsets for the task were also explored. Most of these features are based on a comparison between domain-specific documents collection and a reference general corpus. In case of the supervised approach, the feature Words Count occurs in all of the subsets, so this feature is useful for the classifier, because

values of other features may have different meanings for single- and multi-word terms.

In cases when one dataset is used for training and another to test, we could not get stable performance gain using machine learning. Even the datasets are of the same field, a distribution of terms can be different. So it is still unclear if it is possible to recognize terms from unseen data of the same field having the once-trained classifier.

For our experiments we implemented the simple method of term candidates extraction: we filter out ngrams that do not match predefined part-of-speech patterns. This step of ATR can be performed in other ways, for example by shallow parsing, or chunking [3], generating patterns from the dataset [17] or recognizing term variants.

Another direction of further research is related to the evaluation of the algorithms on more datasets of different languages and researching the ability of cross-domain term recognition, i.e. using a dataset of one domain to recognize terms from others.

Also of particular interest is the implementation and evaluation of semi- and unsupervised methods that involve machine learning techniques.

References

[1] K. Ahmad, L. Gillam, L. Tostevin, et al. University of surrey participation in trec8: Weirdness indexing for logical document extrapolation and retrieval (wilder). In *The Eighth Text REtrieval Conference (TREC-8)*, 1999.

[2] Lars Ahrenberg. Term extraction: A review draft version 091221. 2009.

[3] K.W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[4] B. Daille. Study and implementation of combined techniques for automatic extraction of terminology. *The balancing act: Combining symbolic and statistical approaches to language*, 1:49–66, 1996.

[5] B. Daille, B. Habert, C. Jacquemin, and J. Royauté. Empirical observation of term variations and principles for their description. *Terminology*, 3(2):197–257, 1996.

[6] B. Dobrov and N. Loukachevitch. Multiple evidence for term extraction in broad domains. In *Proceedings of the 8th Recent Advances in Natural Language Processing Conference (RANLP 2011). Hissar, Bulgaria*, pages 710–715, 2011.

[7] J. Foo. Term extraction using machine learning. 2009.

[3] Free chunker can be found in OpenNLP project: http://opennlp.apache.org

[8] K.T. Frantzi and S. Ananiadou. Extracting nested collocations. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 41–46. Association for Computational Linguistics, 1996.

[9] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[10] K. Kageura and B. Umino. Methods of automatic term recognition: A review. *Terminology*, 3(2):259–289, 1996.

[11] C.D. Manning and P. Raghavan. *Introduction to information retrieval*, volume 1.

[12] C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[13] L.C. Molina, L. Belanche, and À. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313. IEEE, 2002.

[14] R. Navigli and P. Velardi. Semantic interpretation of terminological strings. In *Proc. 6th Intl Conf. Terminology and Knowledge Eng*, pages 95–100, 2002.

[15] M.A. Nokel, E.I. Bolshakova, and N.V. Loukachevitch. Combining multiple features for single- word term extraction. 2012.

[16] Y. Park, R.J. Byrd, and B.K. Boguraev. Automatic glossary extraction: beyond terminology identification. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[17] A. Patry and P. Langlais. Corpus-based terminology extraction. In *Terminology and Content Development–Proceedings of 7th International Conference On Terminology and Knowledge Engineering, Litera, Copenhagen*, 2005.

[18] M. Pazienza, M. Pennacchiotti, and F. Zanzotto. Terminology extraction: an analysis of linguistic and statistical approaches. *Knowledge Mining*, pages 255–279, 2005.

[19] A. Peñas, F. Verdejo, J. Gonzalo, et al. Corpus-based terminology extraction applied to information access. In *Proceedings of Corpus Linguistics*, volume 2001. Citeseer, 2001.

[20] F. Sclano and P. Velardi. Termextractor: a web application to learn the shared terminology of emergent web communities. *Enterprise Interoperability II*, pages 287–290, 2007.

[21] P. Velardi, M. Missikoff, and R. Basili. Identification of relevant terms to support the construction of domain ontologies. In *Proceedings of the workshop*

on Human Language Technology and Knowledge Management-Volume 2001, page 5. Association for Computational Linguistics, 2001.

[22] Y. Yang, H. Yu, Y. Meng, Y. Lu, and Y. Xia. Fault-tolerant learning for term extraction. 2011.

[23] W. Zhang, T. Yoshida, and X. Tang. Using ontology to improve precision of terminology extraction from documents. *Expert Systems with Applications*, 36(5):9333–9339, 2009.

[24] Ziqi Zhang, Christopher Brewster, and Fabio Ciravegna. A comparative evaluation of term recognition algorithms. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC08), Marrakech, Morocco*, 2008.

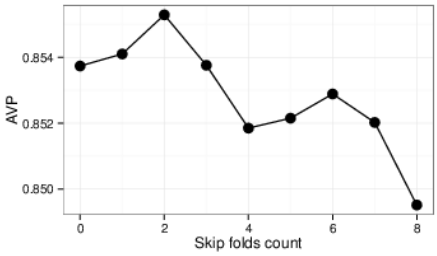

Figure 4: Dependency of AvP from number of excluded folds with fixed testset size: 10-fold cross-validation with 1 test fold and 9 to 1 train folds: Top-100 terms

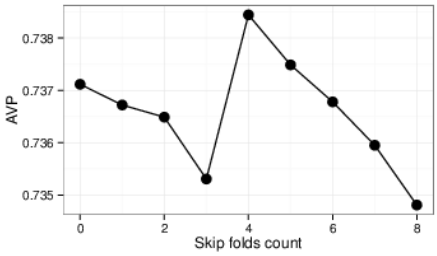

Figure 5: Dependency of AvP from number of excluded folds with fixed testset size: Top-1000 terms

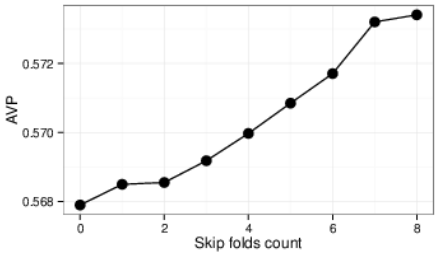

Figure 6: Dependency of AvP from number of excluded folds with fixed testset size: Top-5000 terms

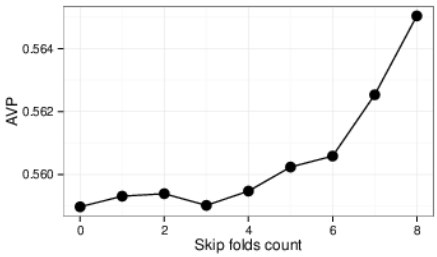

Figure 7: Dependency of AvP from number of excluded folds with fixed testset size: Top-10000 terms

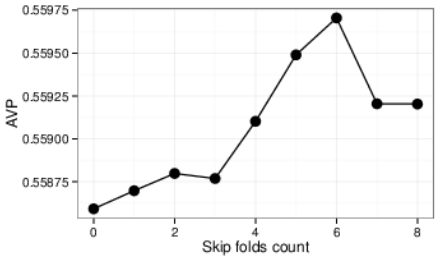

Figure 8: Dependency of AvP from number of excluded folds with fixed testset size: Top-20000 terms

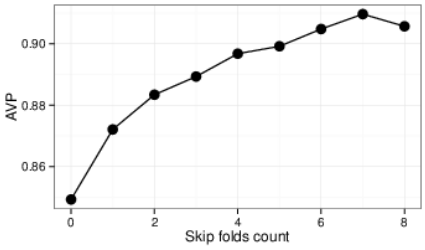

Figure 9: Dependency of AvP from number of excluded folds with changing testset size: 10-fold cross-validation with 1 to 9 test folds and 9 to 1 train folds: Top-100 terms

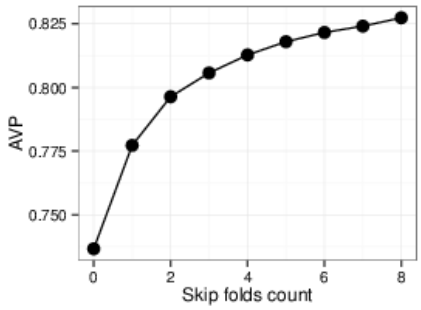

Figure 10: Dependency of AvP from number of excluded folds with changing testset size: Top-1000 terms

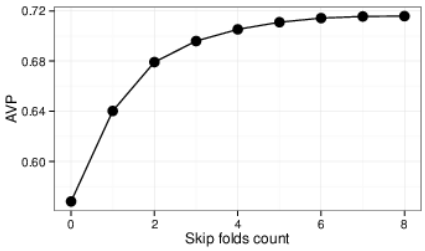

Figure 11: Dependency of AvP from number of excluded folds with changing testset size: Top-5000 terms

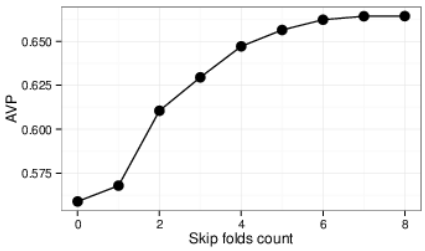

Figure 12: Dependency of AvP from number of excluded folds with changing testset size: Top-10000 terms

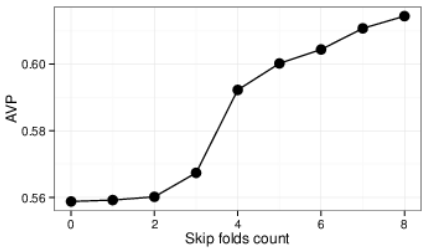

Figure 13: Dependency of AvP from number of excluded folds with changing testset size: Top-20000 terms