Combining Named Entity Recognition Methods for Concept Extraction in Microposts

Štefan Dlugolinský
upsysdlu@savba.sk

Peter Krammer
upsypkra@savba.sk

Marek Ciglan
upsymaci@savba.sk

Michal Laclavík
laclavik.ui@savba.sk

Ladislav Hluchý
upsylhlu@savba.sk

Institute of Informatics, Slovak Academy of Sciences
Dúbravská cesta 9
845 07 Bratislava, Slovakia

ABSTRACT

NER in microposts is a key and challenging task of mining semantics from social media. Our evaluation of a number of popular NE recognizers over a micropost dataset has shown a significant drop-off in results quality. Current state-of-the-art NER methods perform much better on formal text than on microposts. However, the experiment provided us with an interesting observation – although individual NER tools did not perform very well on micropost data, we have received recall over 90% when we merged all the results of the examined tools. This means that if we would be able to combine different NE recognizers in a meaningful way, we might be able to get NER in microposts of an acceptable quality. In this paper, we propose a method for NER in microposts, which is designed to combine annotations yielded by existing NER tools in order to produce more precise results than input tools alone. We combine NE recognizers utilizing ML techniques, namely decision tree and random forest using the C4.5 algorithm. The main advantage of the proposed method lies in the possibility of combining arbitrary NER methods and in its application on short, informal texts. The evaluation on a standard dataset shows that the proposed approach outperforms underlying NER methods as well as a baseline recognizer, which is a simple combination of the best underlying recognizers for each target NE class. To the best of our knowledge, up-to-date, the proposed approach achieves the highest F_1 score on the #MSM2013 dataset.

Categories and Subject Descriptors

1.2.7 [**Natural language processing**]: Language parsing and understanding, Text analysis.

Keywords

named entity recognition, machine learning, microposts

Published as part of the #Microposts2014 Workshop proceedings, available online as CEUR Vol-1141 (http://ceur-ws.org/Vol-1141)

#Microposts2014, April 7th, 2014, Seoul, Korea.

1. INTRODUCTION

A significant growth of social media interaction can be observed in recent years. People are able to interact through the Internet from almost anywhere at anytime. They can share their experience, thoughts and knowledge instantly and they do it in mass dimensions. The easiest and probably the most popular way of interaction on the Web is through microposts – short text messages posted on the Web. There is a plenty of services offering such communication, notorious examples of microposts include tweets, Facebook statuses, comments, Google+ posts, Instagram photos. Microposts analysis has a big potential in hidden knowledge that can be used in wide range of domains like emergency response, public opinion assessment, business or political sentiment analysis and many more. The most important task in order to analyze and make sense of microposts is the Named Entity Recognition (NER). NER in microposts is a challenging problem because of a limited size of a single micropost, prevalence of term ambiguity, noisy content, multilingualism [2]. These are the main reasons why existing NER methods perform better on formal newswire text than on microposts and there is clearly a space for new methods of NER designed for social media streams.

In this paper, we first evaluate multiple popular and widely used NER methods on the micropost data. The results show a significant decrease of result quality compared to those reported for newswire texts. An interesting observation from the experiment is that we can achieve recall over 90% on the micropost data, when all the results are unified. This means, that in theory, we could achieve very high quality annotations of named entities (NEs) in microposts just by combining existing NER tools in a "smart" way. The rest of the paper is dedicated to the research question, how to combine annotations of different NER tools in order to achieve better recognition in microposts.

We propose an approach for combining NER methods represented by different NE recognizers in order to make a new NE recognizer intended to be used on microposts. The method is designed to combine annotations produced by different NER tools by exploiting machine learning (ML) techniques. We use the term annotation to refer to a substring of an input text that has been marked by a NER tool as a reference to an entity of one of target classes; i.e., LOC, MISC, ORG and PER. The main challenge is the transformation of text annotations produced by NER tools into a

form usable for training ML classification algorithms. Once the NER annotations were transformed to an appropriate format, we have performed an evaluation of a number of popular ML classification techniques. The best performing on our problem domain was the C4.5 algorithm [15] that was used to train decision tree (DT) and random forest (RF) models. The resulting classification model outperformed the best of underlying individual recognizers by more than 10% in F_1 score and a chosen baseline model by 3% in F_1 score.

The main contributions of the work are following: (i) We show that although existing NER tools designed for news text do not perform well on microposts, by merging results of several different NER tools, we can achieve high recall and precision. (ii) We utilize ML classifiers to combine the outputs of multiple NE recognizers. The principal challenge is the transformation of text annotations yielded by NER tools to feature vectors that can be used for the training of classification algorithms. (iii) We provide an extensive evaluation of popular classification models to asses their suitability for the problem of combing results of NER tools. For the best performing ones, we have studied the influence of algorithms parameters on the classification results.

The paper is structured as follows. In Section 2, we briefly summarize research works related to NER. In Section 3 we conduct an experiment, in which a number of existing popular NER tools are evaluated on microposts data. Results show dramatic drop in quality measures compared to the numbers reported on news datasets. In Section 4, we define a baseline NE recognizer, explain our approach of combining NER tools and evaluate our NE recognition models. Finally, Section 5 discusses open issues and Section 6 summarizes our results and concludes the paper.

2. RELATED WORK

There has been a large amount of NER research conducted on formal text, such as newswire or biomedical text. The performance of NE recognizers for this kind of text is comparable to that of humans. For instance, the MUC-7 NE task, where the best NE recognizer scored $F_1 = 93.39\%$, while the annotators scored $F_1 = 97.60\%$ and $F_1 = 96.95\%$ [13]. Another example is CoNLL-2003 shared task, where the best NER recognizer scored $F_1 = 88.76\%$ in English test [22]. It has been later outperformed by Ratinov and Roth [17] achieving $F_1 = 90.8\%$. NE recognizers, which have been designed for these tasks and which achieve state-of-the-art performance results, heavily rely on linguistic features observable in formal text. But many of the important features absent in microposts; e.g. capitalization. Therefore, news-trained recognizers perform worse on them. The performance drop-off is also caused by nature of microposts content – its length, informality, noise and multilingualism. Many of the problems related to NER in microposts are discussed by Bontcheva and Rout in [2].

The idea of combining different methods for NER is not new. It has been successfully applied on formal text by Florian et al. [8], who combine four diverse classifying methods; i.e., transformation-based learning, hidden Markov model, robust risk minimization (RRM) and maximum entropy. Classifiers are complemented by gazetteers together with the output of two externally trained NE recognizers and the whole is used to extract text features. The RMM method is used in order to select a good performing combination of the features. Todorovski and Džeroski [23] introduce meta decision trees (MDT) for combining multiple classifiers. They present a C4.5 algorithm-based training algorithm for producing MDTs. Another application is by Si et al. [21], who combine several NER methods for bio-entity recognition in biomedical texts. They experiment with combining NE classifiers by three different approaches; i.e., majority vote, unstructured exponential model and conditional random field. Also Saha and Ekbal [20] use seven diverse NER classifiers to build a number of voting models depending upon identified text features that are selected mostly without a domain knowledge.

Regarding the NER for tweets, there is also a similar approach taken by Liu et al. [12]. Authors combine a k-Nearest Neighbors (k-NN) classifier with a linear Conditional Random Fields (CRF) model under a semi-supervised learning framework and show increase in F_1 with respect to a baseline system, which is its modified version without k-NN and semi-supervised learning. Etter et al. [6] deal with multilingual NER for short informal text. They do not rely on language dependent features such as dictionaries or POS tagging, but they use language independent features derived from the character composition of a word and its context in a message; i.e., words, character n-grams for words, $\pm k$ words to the left, message length, word length and word position in message. They use an algorithm that combines Support Vector Machine (SVM) with a Hidden Markov Model (HMM) to train a NER model on a manually annotated data. The experiments show that the language independent features lead to F_1 score increase and the model outperforms Ritter et al. [19]. Ritter et al. [19] present re-built NLP pipeline for tweets; i.e., POS tagger, chunker and NE recognizer. The NE recognizer leverages the redundancy inherent in tweets using Labeled LDA [16] to exploit Freebase[1] dictionaries as a source of distant supervision. TwiNER, a novel unsupervised NER system for targeted tweet streams is proposed by Li et al. [11]. Similarly to Etter et al. [6], TwiNER does not rely on any linguistic features of the text. It aggregates information garnered from the Web and Wikipedia. The advantage of TwiNER is that it does not require manually annotated training set. On the other hand, TwiNER does not categorize the type of discovered NEs. Authors prefer the problem of correctly locating and recognizing presence of NEs instead of their classification. Habib and Keulen [9], the winning solution of the #MSM2013 IE Challenge, splits the NER problem in named entity extraction (NEE) and named entity classification (NEC), too. The NEE task is performed by union of entities recognized by two models; i.e., CRF and SVM. Both models are trained on manually labeled tweet data. The CRF involves POS tags and capitalization of the words as features. The SVM segments tweet using Li et al. [11] approach and enriches the segments by external knowledge base (KB). It uses the same features as the SVM model and information from external KB.

3. COMBINED NER METHODS

We have used state-of-the-art NER methods represented by various existing NE recognizers. These methods were combined in our classification models discussed later in this paper. Below we briefly describe used NE recognizers focusing on their NER methods.

1) ANNIE (v7.1) [4] relies on finite state algorithms,

[1] `http://www.freebase.com`

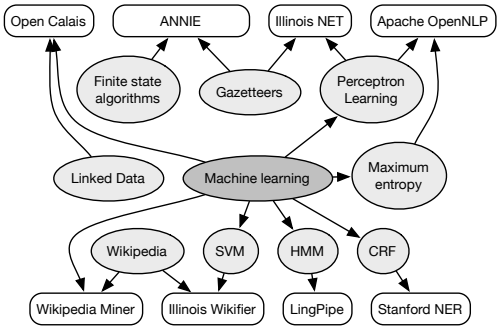

Figure 1: Outline of NE recognizers

gazetteers and the JAPE (Java Annotation Patterns Engine) language. *2) Apache OpenNLP*[2] (v1.5.2) is based on maximum entropy models and perceptron learning algorithm. *3) Illinois Named Entity Tagger* (v1.0.4) [17] uses a regularized averaged perceptron with external knowledge (unlabeled text, gazetteers built from Wikipedia and word class models). We have used Illinois NET with 4-label type set and default configuration. *4) Illinois Wikifier* (v1.0[3]) [18] is based on a Ranking SVM and exploits Wikipedia link structure in disambiguation. *5) Open Calais* operates behind a shroud of mystery since there is not much information available about how its NE recognition works. Official sources[4] say, that it uses NLP, ML and other methods as well as Linked Data. *6) Stanford Named Entity Recognizer* (v1.2.7) [7] is based on CRF sequence models. We have used the English 4-class caseless CONLL model[5]. *7) Wikipedia Miner*[6] [14] is a text annotation tool, which is capable of annotating Wikipedia topics in a given text. It exploits Wikipedia link graph, Wikipedia category hierarchy and relies on ML classifiers, which are used for measuring relatedness of concepts and terms, as well as for measuring disambiguation. We have applied this software to discover Wikipedia topics, which were then tagged according to the DBPedia Ontology[7].

Most of the NE recognizers are based on statistical learning methods. Some of them use also gazetteers and other external knowledge like Wikipedia or Linked Data. Outline of the NE recognizers is depicted in Figure 1.

3.1 NE Recognizers Evaluation

In this section, we evaluate NER methods described in Section 3 on a micropost data corpus. Our intent was to see the performance of each individual NE recognizer. The evaluation was focused also on analysis, which NE recognizer is more suitable for particular named entity class and whether NE recognizers produce diverse results. NE recognizers were evaluated over the adapted #MSM2013 IE Challenge training dataset [1]. We have taken the 1.5 version and cleaned it from duplicate as well as from overlapping microposts with the test dataset. The cleaned training dataset

[2] `http://opennlp.apache.org`
[3] `http://cogcomp.cs.illinois.edu/page/download_view/Wikifier`
[4] `http://www.opencalais.com/about`
[5] english.conll.4class.caseless.distsim.crf.ser.gz
[6] `http://wikipedia-miner.cms.waikato.ac.nz`
[7] `http://dbpedia.org/Ontology`

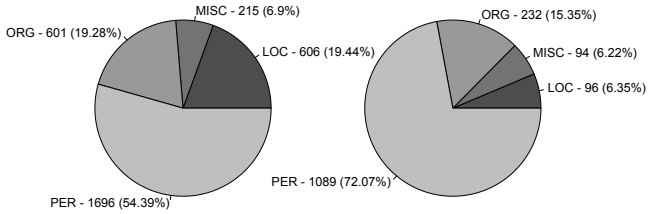

Figure 2: Named entity occurrences in train (left) and test (right) datasets

finally contained 2752 unique manually annotated microposts with classification restricted to four entity types: PER, LOC, ORG and MISC. We have also adapted a test dataset from the #MSM2013 IE Challenge on which we later evaluated our classification models. The occurrence of NEs in both datasets is displayed in Figure 2. Named entity types were not equally distributed. The most frequent entity type in both datasets was PER and the least frequent was MISC. Datasets used in this paper are also available for download[8] in GATE SerialDataStore format. Datasets includes results of all the used NE recognizers as well as our NER models discussed later in the paper.

Evaluated NE recognizers were not specially configured, tweaked or trained for microposts prior to the evaluation. We wanted to see, how they cope with the different kind of text that they were trained for. The alignment with our taxonomy was done by simple mapping. Evaluation results are displayed in Table 1 and ordered by Micro avg. F_1 score. We provide also a Macro summary which averages P, R and F_1 measures on a per document basis, while the Micro summary considers the whole dataset as a one document. The evaluation has also shown, that the NE recognizers produced diverse annotations. This behavior could be seen in raised recall after the results were unified and cleaned from duplicates. Figure 3 illustrates the situation and the possible recall, which could be theoretically achieved when combining the recognizers.

More details about the evaluation can be found in [5]. Some of the evaluation results may slightly differ from those displayed in Table 1. It is because we did accept adjectivals and demonymic forms for countries as *MISC* type in this work; e.g., Americans, English.

4. COMBINING NE RECOGNIZERS

The idea of how to combine NE recognizers was to use ML techniques to build a classification model, which would

[8] `http://ikt.ui.sav.sk/microposts/`

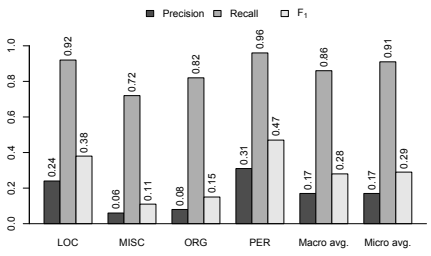

Figure 3: Precision, Recall and F_1 of unified NE recognizers

Table 1: Evaluation of NE recognizers over the training dataset

	F_1				Macro avg.			Micro avg.		
NE recognizer	LOC	MISC	ORG	PER	P	R	F_1	P	R	F_1
OpenCalais	**0.74**	0.26	**0.56**	0.69	0.66	0.50	**0.56**	**0.72**	0.60	**0.65**
Illinois NET	0.72	0.14	0.36	**0.79**	0.50	0.51	0.50	0.61	**0.65**	0.63
Stanford NER	0.67	0.11	0.29	0.75	0.46	0.45	0.46	0.60	0.59	0.60
ANNIE	0.68	–	0.36	0.61	0.71	0.37	0.41	0.64	0.48	0.55
Illinois Wikifier	0.55	0.16	0.51	0.62	0.54	0.42	0.46	0.62	0.47	0.54
Apache OpenNLP	0.51	–	0.27	0.58	0.68	0.28	0.34	0.62	0.38	0.47
Wikipedia Miner	0.56	0.06	0.33	0.61	0.34	**0.52**	0.39	0.32	0.57	0.41
LingPipe	0.35	–	0.07	0.35	0.40	0.30	0.19	0.16	0.38	0.23
Miscinator	–	**0.46**	–	–	**0.92**	0.09	0.12	0.69	0.03	0.05

be trained on features describing microposts' text as well as annotations produced by involved NE recognizers. We have used the training dataset for building the model and the test dataset for evaluating it and comparing with other NE recognizers (Section 3.1).

According to the evaluation results in Section 3.1, we have chosen seven out of eight NE recognizers based on different methods. The discarded one was LingPipe because of its weak[9] performance on micropost data. Chosen NE recognizers were then complemented by *Miscinator*, an NE recognizer specially designed for the #MSM2013 IE Challenge [24].

As overall recall of the underlying NE recognizers was relatively high, we wanted to gain maximum precision while not devalue the recall. We decided to involve ML techniques, but it was necessary to transform this problem into a standard ML task. In this case it was suitable to transform the task of NER into a task of classification. The intent was that ML process would produce a classification model capable of classifying given annotations from involved methods into four target classes LOC, MISC, ORG, PER and one special class NULL indicating that the annotation did not belong to any of the four target classes. Then a simple algorithm would be applied to merge the re-classified annotations into final results.

4.1 Baseline NE Recognizer

We have defined a baseline NE recognizer in the way that each target entity class was extracted by the best NE recognizer according to the evaluation made over the training dataset (section 3.1); i.e., LOC, MISC and ORG classes were extracted by OpenCalais and PER class was extracted by Illinois NET. The performance of the baseline can be seen in Table 2 together with performances of the NE recognizers considered for combining. The evaluation has been made over the test dataset. We can see that the baseline NE recognizer had outperformed underlying NE recognizers in precision and F_1 measure, which was expected. Our goal was to overcome the performance of the baseline NE recognizer with a model produced by ML approach.

4.2 Transforming NEs into Feature Vectors

We have taken an approach of describing how particular methods performed on different entity types compared to the response of other methods and a manual annotation. Used as a training vector, this description was an input for training a classification model. A vector of input training features was generated for each annotation found by underlying NER methods restricted to following types: LOC, MISC, ORG, PER, NP – noun phrase, VP – verb phrase, OTHER – different type. We called this annotation a reference annotation. The vector of each reference annotation consisted of several sub-vectors (Figure 4).

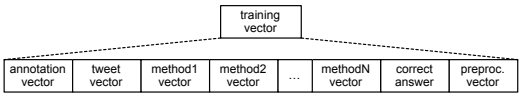

Figure 4: Training vector

The first sub-vector of the training vector was an annotation vector (Figure 5). The annotation vector described the reference annotation – whether it was upper or lower case, used a capital first letter or capitalized all of its words, the word count, and the type of the detected annotation.

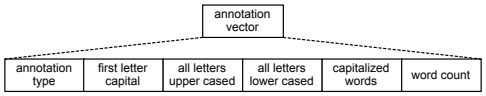

Figure 5: Annotation vector

The second sub-vector described microposts as a whole (Figure 6). It contained features describing whether all words longer than four characters were capitalized, uppercase, or lowercase. We called this sub-vector tweet vector.

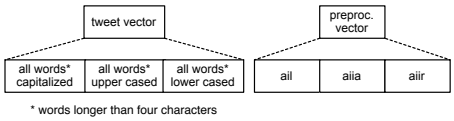

Figure 6: Tweet vector (left) and preprocessing vector (right)

The rest of the sub-vectors were computed according to the overlap of the reference annotation with annotations produced by particular NER method. Such sub-vector (termed a method vector by us) was computed for each method and contained four other vectors describing the overlap of method annotations with reference annotation on each target entity type (Figure 7). The *annotation type* attribute was filled with a class of method annotation that exactly matched position of the reference annotation and was one of the target entity classes, otherwise it was left blank.

Each overlap vector of a particular method and NE class (Figure 8) consisted of five components – *ail*: the average intersection length of a reference annotation with the method

Table 2: Evaluation of NE recognizers over the test dataset

	F_1				Macro avg.			Micro avg.		
Model	LOC	MISC	ORG	PER	P	R	F_1	P	R	F_1
Baseline	**0.61**	**0.29**	0.30	**0.84**	0.69	0.44	**0.51**	**0.83**	0.67	**0.74**
Illinois NET	0.50	0.06	0.32	**0.84**	0.41	**0.46**	0.43	0.65	**0.69**	0.67
Stanford NER	0.51	0.00	0.30	0.82	0.39	0.43	0.41	0.67	0.67	0.67
Open Calais	**0.61**	**0.29**	0.30	0.69	0.64	0.41	0.47	0.66	0.60	0.63
ANNIE	0.48	–	0.19	0.68	0.61	0.32	0.34	0.63	0.52	0.57
Illinois Wikifier	0.34	0.09	**0.46**	0.68	0.44	0.38	0.39	0.63	0.50	0.55
Apache OpenNLP	0.38	–	0.13	0.64	0.57	0.27	0.29	0.62	0.43	0.51
Wikipedia Miner	0.29	0.04	0.29	0.67	0.28	**0.46**	0.32	0.32	0.57	0.41
LingPipe	0.15	–	0.05	0.38	0.37	0.28	0.14	0.15	0.38	0.21
Miscinator	–	0.19	–	–	**0.88**	0.03	0.05	0.52	0.01	0.01

[9] we have used the *English News: MUC-6* model

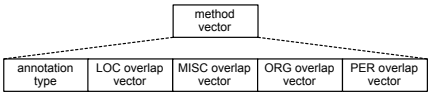

Figure 7: Method vector

annotations of the same NE class, *aiia*: the average intersection ratio of the method annotations of the same NE class with reference annotation, *aiir*: the average intersection ratio of a reference annotation with method annotations of the same NE class, *average confidence* (if the underlying method return such value), and *variance of the average confidence*.

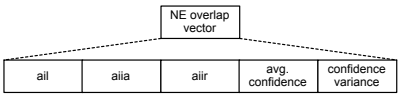

Figure 8: Overlap vector

The *ail* component in overlap vector was computed using formula (1), where R was a fixed reference annotation and M_C was a set of n method annotations of class C intersecting with the reference annotation R. The *ail* component was a simple arithmetic mean of intersection lengths.

$$ail_{(R,M_C)} = \frac{1}{n}\sum_{i=1}^{n}|R \cap M_{Ci}| \quad (1)$$

The *aiia* component was computed using formula (2), which was also a simple arithmetic mean, but the intersection lengths were normalized by lengths of particular method annotations M_{Ci} intersecting with the reference annotation R. We wanted the value of *aiia* component to describe how much were method annotations covered by the reference annotation.

$$aiia_{(R,M_C)} = \frac{1}{n}\sum_{i=1}^{n}\frac{|R \cap M_{Ci}|}{|M_{Ci}|} \quad (2)$$

Similarly, the *aiir* component was computed using formula (3), but the intersection lengths were normalized by length of the reference annotation R. The value of *aiir* component was used to describe how much was the reference annotation covered by method annotations.

$$aiir_{(R,M_C)} = \frac{1}{n}\sum_{i=1}^{n}\frac{|R \cap M_{Ci}|}{|R|} \quad (3)$$

A simple example of overlap vector computation is depicted in Figure 9. The overlap vector is computed for method 4 and PER class according to the highlighted reference annotation. In this example, the reference annotation is M2.PER1, but it can be any method annotation or manual annotation. The rest of the method 4 overlap vectors are zero-valued since method 4 does not return annotations of types LOC, MISC and ORG. Similarly, there will be overlap vectors according to the same reference annotation computed for methods 1, 2 and 3 to finally have all method vectors computed in a training vector. In addition, there will be eight training vectors computed, because of eight annotations taken as reference annotations, where also the manual annotation PER is included.

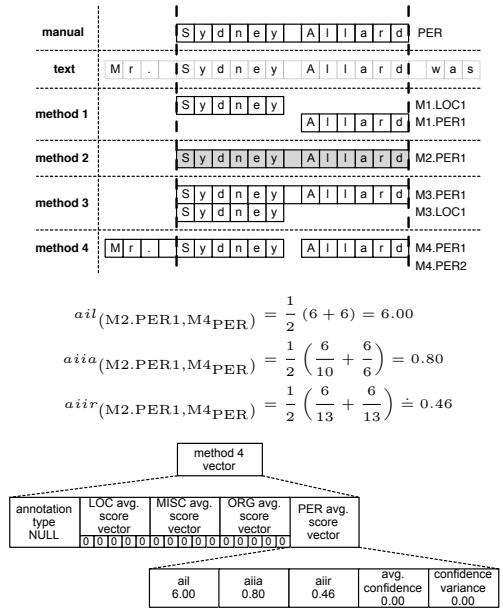

Figure 9: Example of overlap vector computation

The last two components in the training vector were the correct answer (i.e., the correct annotation type taken from manual annotation) and a special preprocessing vector (Figure 6). The preprocessing vector included three components: *ail*, *aiia* and *aiir*, which described the intersection of the reference annotation when it was correct with the correct answer. If the reference annotation was not correct the values of the preprocessing vector components were set to zero.

The number of learning features depended on the number of combined methods, since for each involved method a new method vector was computed and included into the training vector. There were some features, which were less or more important or not important at all. The effect of specific learning features is discussed later.

4.3 Training Data Preprocessing

Training data was generated automatically as a collection of training vectors, which needed further processing prior to apply ML algorithms. There have been duplicate training vectors removed in order to eliminate distortion in training and validation process thus getting a more balanced classification model.

According to the preprocessing vector (Figure 6), there have been training vectors removed, in which the *annotation type* attribute in the *annotation vector* was correct but the *aiir* attribute in the preprocessing vector was not equal to 1.0, i.e., the bounds of the reference annotation were not equal to the bounds of the correct answer. In previous versions, we tried to accept all the training vectors whose *aiir* attribute was at least 0.95, i.e., the reference annotation overlapped with the correct answer at least on 95%, but this led to models with lower precision.

We have removed also several attributes, which led to zero information gain and which were not useful for the classification, i.e., attributes with the same value for all the training vectors. They were usually *average confidence* and *variance of the average confidence* scores, because some NE recog-

Table 3: Performance of classification models built by different algorithms

Model	AUROC	ACC	F_1
Decision Tree J48	**0.939**	0.969	**0.938**
Random Forest	0.927	**0.972**	0.925
Bagging	0.912	**0.972**	0.908
Multilayer Perceptron	0.895	0.955	0.890
Dagging	0.889	0.922	0.880
Bayess Net	0.857	0.954	0.865
RBF Network	0.850	0.923	0.835
AdaBoost.M1	0.811	0.804	0.750
Naive Bayes	0.797	0.919	0.814

nizers did not provide annotation confidence information, hence both attributes were always zero and therefore also their information gain. Due to same reasons, we have removed also attributes, which contained information in less than 3% of records. Attributes of the *preprocessing vector* have been also removed.

The preprocessing phase had significantly reduced the size of training data and therefore memory requirements as well as it had sped up the training process. It started with a set of $\sim 63,000$ training vectors with ~ 200 attributes and finished on $\sim 31,000$ unique records with ~ 100 highly relevant attributes.

4.4 Model Training and Evaluation

We have tried several algorithms to train different classification model candidates, which we compared according to the F_1 score. We have also examined AUROC and ACC (accuracy) measures. All these three measures were obtained from 10-fold cross validation of the model candidates over the training dataset. Cross validation served as a good method for identifying suitable model candidates, because it avoided an effect of overfitting without a need of another test dataset. The best performance has been achieved by DT classification model built with J48[10] algorithm (DTJ48) followed by RF [3] model. The third was a classification model based on REPTree (Reduced Error Pruned Tree) built with Bagging algorithm (Table 3). We have focused on the first two best performing algorithms and built several classification models while varying some of input parameters of these algorithms in order to gain precision and recall. It was *Minimum Number of Instances per Leaf* parameter (hereinafter parameter "M") for DTJ48 and *number of trees* for RF. The classification models were evaluated using a hold-out validation method over the test dataset. Evaluation results are displayed in Table 4. The best performing were models based on RF, which outperformed models based on DT, baseline recognizer and all the underlying NE recognizers. We can see that recall and precision have been growing with the number of trees in the RF models and continued to converge to 79% and 76% respectively. This behavior is more obvious in Figure 10, where F_1 measures are depicted for particular NE classes according to the variated number of trees. Dashed lines indicate score of the baseline model.

Evaluation results of models built with J48 algorithm (C4.5), while varying the M parameter, are displayed in Figure 11. We can see that the F_1 score for LOC has been approaching the baseline score similarly as it was for RF algorithm while varying the number of trees parameter. Analogous behavior can be seen in Macro and Micro average scores. In ORG and PER classification the score was higher

[10]J48 is an implementation of C4.5 algorithm

Table 4: Evaluation of classification models over the test dataset

	F_1				Macro avg.			Micro avg.		
Model	LOC	MISC	ORG	PER	P	R	F_1	P	R	F_1
RF N400	0.60	0.23	**0.49**	**0.88**	0.60	0.53	**0.55**	0.79	**0.76**	**0.77**
RF N300	0.60	0.23	**0.49**	**0.88**	0.60	0.53	**0.55**	0.79	**0.76**	**0.77**
RF N200	0.59	0.24	0.48	**0.88**	0.60	0.53	**0.55**	0.79	**0.76**	**0.77**
RF N100	0.58	0.23	0.48	**0.88**	0.59	0.53	0.54	0.79	**0.76**	**0.77**
RF N9	0.57	0.26	0.47	0.87	0.55	**0.54**	0.54	0.76	**0.76**	0.76
RF N21	0.55	0.26	0.47	0.87	0.56	0.53	0.54	0.77	**0.76**	0.76
RF N17	0.56	0.26	0.48	**0.88**	0.56	**0.54**	0.54	0.77	**0.76**	0.76
RF N14	0.55	0.26	0.46	**0.88**	0.55	0.53	0.54	0.76	**0.76**	0.76
RF N11	0.57	0.25	0.46	0.87	0.56	0.53	0.54	0.76	**0.76**	0.76
DTJ48 M13	0.57	**0.36**	0.36	0.87	0.60	0.52	0.54	0.78	0.73	0.75
RF N7	0.56	0.25	0.44	0.87	0.53	0.53	0.53	0.75	**0.76**	0.75
DTJ48 M11	0.59	0.27	0.40	0.86	0.56	0.51	0.53	0.77	0.73	0.75
DTJ48 M9	0.55	0.29	0.39	0.86	0.55	0.51	0.52	0.77	0.72	0.75
DTJ48 M7	0.57	0.23	0.41	0.86	0.53	0.51	0.52	0.75	0.73	0.74
RF N5	0.53	0.22	0.42	0.86	0.51	0.52	0.51	0.73	0.75	0.74
Baseline	**0.61**	0.29	0.30	0.84	**0.69**	0.44	0.51	**0.83**	0.67	0.74
DTJ48 M5	0.54	0.23	0.43	0.85	0.53	0.51	0.51	0.75	0.72	0.74
#MSM2013 21_3	0.50	0.31	0.41	0.83	0.51	0.53	0.51	0.70	0.73	0.71
DTJ48 M2	0.45	0.31	0.37	0.84	0.50	0.49	0.49	0.71	0.71	0.71
RF N3	0.50	0.20	0.37	0.85	0.46	0.50	0.48	0.68	0.73	0.71
RF N2	0.51	0.15	0.33	0.84	0.44	0.49	0.46	0.64	0.71	0.68

than the baseline or at least the same. We cannot say, that it has been growing with the parameter M. The same applies for MISC, where the F_1 score varied around the baseline. In general, increasing minimum number of instances per leaf in DT (parameter M) led to models with higher recall and precision. There were four classification models, which have slightly outperformed the baseline model, but not as much as the RF models.

The #MSM2013 21_3 model in the Table 4 is our submission to the #MSM2013 IE Challenge [24]. This model was one of our early models, which were based on groundwork of this paper. The model has finished on the second place in the challenge loosing 1% in F_1 on a winner Habib et. al [9]. Results of this model in the table may be slightly worse than the official challenge results[11], since we have used more strict evaluation criteria. We did not accept partially correct consecutive annotations; i.e., PER/Christian PER/Bale was incorrect, while PER/Christian Bale was correct. For a better

[11]`http://oak.dcs.shef.ac.uk/msm2013/ie_challenge/results/challenge_results_summary.pdf`

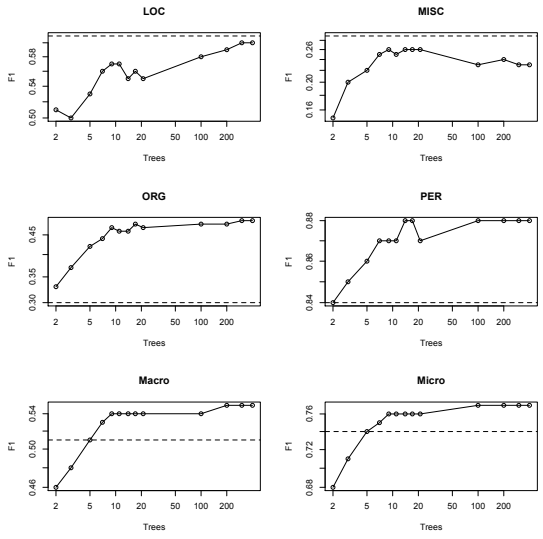

Figure 10: Impact on F_1 while varying number of trees for Random Forest algorithm

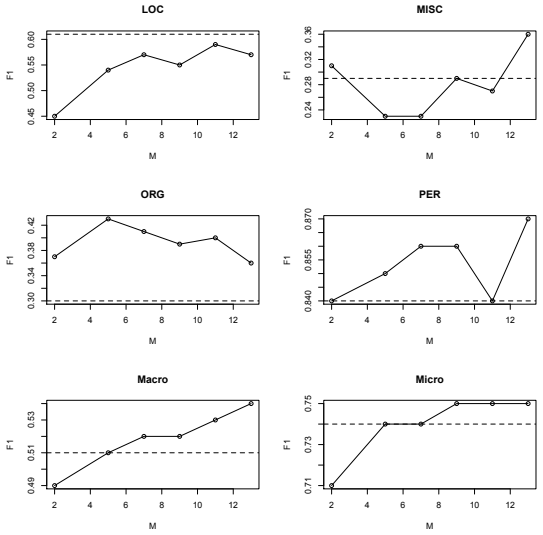

Figure 11: Impact on F_1 while varying parameter M for Decision Tree J48 (C4.5) algorithm

comparison of the models, we present precision, recall and F_1 measures of the best performing model – RF N400, best DT model – DTJ48 M13, baseline recognizer and the three best performing NE recognizers in Figure 12. The gain in precision of the RF N400 model with respect to the NE recognizer with the highest precision – Stanford NER was 18%. However, the baseline recognizer had higher precision than RF N400 by 4%. Model based on DT – DTJ48 M13 was the third best in precision followed by Stanford NER. The highest score in recall among the combined NE recognizers has been achieved by Illinois NET reaching 69%. The gain in recall of the RF N400 model with respect to Illinois NET was 10%. RF N400 reached the highest score in recall followed by DTJ48 M13 and Illinois NET. Stanford NER and the baseline recognizer shared the fourth place.

The highest score in F_1 measure among the combined NE recognizers has been achieved by Illinois NET and Stanford NER, which both reached 67%. The gain in F_1 of RF N400 with respect to them was 15%. RF N400 model with 400 trees has outperformed also the second DTJ48 M13 model and the third baseline recognizer, whose gain was 10%. A comparison on NE class basis is depicted in Figure 13. We did not include the baseline recognizer in the charts, since it is represented there by its NE recognizers (see Section 4.1). Our RF N400 model was the best in recognizing two most occurring entity classes in the test dataset – ORG and PER. It has gained 7% and 5% with respect to Illinois Wikifier and Illinois NET respectively. The best in recognizing LOC entities was Open Calais, on which the RF N400 model lost 1%. The MISC entity type was a domain of the DTJ48 M13 model, which has gained 24% with respect to the second Open Calais.

Closer analysis of annotation results has shown, that there have been many results correctly classified, but they did not exactly match position in text; i.e., results were partially correct. Therefore we tried to apply post-processing and trimmed non-alphabetical characters off the results. We have also removed definite articles from LOC and PER results. Moreover, we have removed titles from PER results; e.g., Dr., Mr. or Sir. Evaluation of models with this simple post-processing (PP) is displayed in Table 5. We have applied post-processing on the best versions of RF and DT models. The gain in F_1 with respect to models without post-processing was 3%. Finally, we tried to build up a model by combining our best models, which were RF N400 PP for LOC, ORG, PER NE classes and DTJ48 M13 PP for MISC class. This model had better performance in MISC recognition, but the overall improvement was not markable, because the occurrence of MISC entities in the test dataset was very low, thus it did not significantly affect the F_1 score.

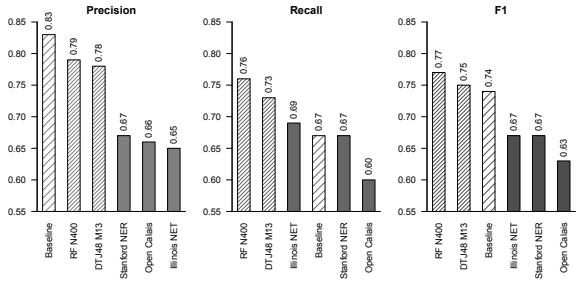

Figure 12: Comparison of the three best NE recognizers with the baseline recognizer and our two best performing models RF N400 and DTJ48 M13

Table 5: Evaluation of classification models using post-processing (PP) over the test dataset

	F_1				Macro avg.			Micro avg.		
Model	LOC	MISC	ORG	PER	P	R	F_1	P	R	F_1
C4.5M13 PP + RF N400 PP	**0.61**	**0.36**	**0.56**	**0.88**	0.66	**0.58**	**0.60**	0.80	**0.78**	**0.79**
RF N400 PP	**0.61**	0.25	**0.56**	**0.88**	0.63	0.55	0.58	0.80	0.77	**0.79**
DTJ48 M13 PP	0.58	**0.36**	0.44	**0.88**	0.63	0.54	0.56	0.80	0.75	0.77
RF N400	0.60	0.23	0.49	**0.88**	0.60	0.53	0.55	0.79	0.76	0.77
DTJ48 M13	0.57	**0.36**	0.36	0.87	0.60	0.52	0.54	0.78	0.73	0.75
Baseline	**0.61**	0.29	0.30	0.84	**0.69**	0.44	0.51	**0.83**	0.67	0.74
#MSM2013 21_3	0.50	0.31	0.41	0.83	0.51	0.53	0.51	0.70	0.73	0.71

5. DISCUSSION AND FUTURE WORK

The structure of the best models (DTJ48 M13 and RF N400) is based on DTs, which use rules always related to one input attribute. This could present a weakness of

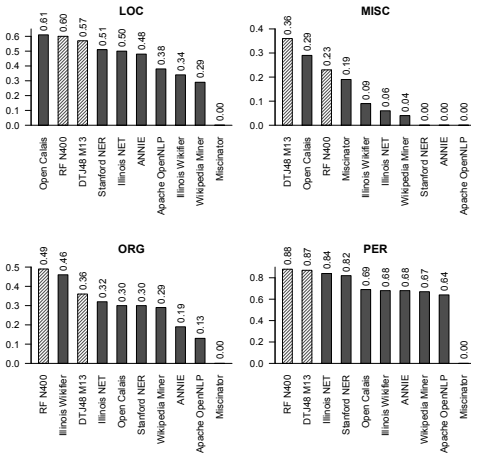

Figure 13: Comparison of the combined NE recognizers with our two best performing models RF N400 and DTJ48 M13 by F_1 and NE class

these models. One possible solution could be to use multivariate DTs, which support multiple attributes per node in a tree and can handle also correlated attributes [10]. The drawback of using multivariate DTs is in the time needed to built them, but on the other hand their time performance is higher, because they do not test the same attribute multiple times. We expect that such models could better utilize the potential of data and therefore could be also more accurate than RF or DT models.

6. CONCLUSIONS

We have shown an approach of combining NE recognizers based on diverse methods on a task of NER in microposts and examined several ML techniques for the combination of text and annotation features produced by the recognizers. The best performing were RF and DT based on C4.5 algorithm. Combination models produced by these algorithms have achieved performance superior to that of underlying NE recognizers as well as the baseline recognizer, which was built of the best performing NE recognizers for each target NE class. The best of our combination models was RF N400, an RF model with 400 trees. Its gain in F_1 with respect to the best individual NE recognizer was 15% and with respect to the baseline recognizer 4%. Performance of the RF and DT models indicated that ML techniques lead to more favorable combination of underlying NE recognizers than it was done manually in the baseline NE recognizer. The advantage of the ML models is that they can adapt to actual text according to its features and annotations from underlying NE recognizers, as well as benefit from given negative examples.

7. ACKNOWLEDGMENTS

This work was supported by projects VEGA 2/0185/13, VENIS FP7-284984 and CLAN APVV-0809-11.

8. REFERENCES

[1] A. E. C. Basave, A. Varga, M. Rowe, M. Stankovic, and A.-S. Dadzie. Making sense of microposts (#msm2013) concept extraction challenge. In *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*, pages 1–15, 2013.

[2] K. Bontcheva and D. Rout. Making sense of social media streams through semantics: a survey. *Semantic Web*, 2012.

[3] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[4] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. ACL'02. ACL, 2002.

[5] S. Dlugolinsky, M. Ciglan, and M. Laclavik. Evaluation of named entity recognition tools on microposts. INES 2013. IEEE, 2013.

[6] D. Etter, F. Ferraro, R. Cotterell, O. Buzek, and B. Van Durme. Nerit: Named entity recognition for informal text. Technical report, Technical Report 11, HLTCE, Johns Hopkins University, July 2013.

[7] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. ACL.

[8] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. CONLL '03, pages 168–171, Stroudsburg, PA, USA, 2003. ACL.

[9] M. Habib, M. V. Keulen, and Z. Zhu. Concept extraction challenge: University of Twente at #msm2013. In *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*, pages 17–20, 2013.

[10] T. S. Korting. C4. 5 algorithm and multivariate decision trees, image processing division. *National Institute for Space Research–INPE São José dos Campos–SP, Brazil*, 2006.

[11] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee. Twiner: Named entity recognition in targeted twitter stream. SIGIR '12, pages 721–730, New York, NY, USA, 2012. ACM.

[12] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. HLT '11, pages 359–367, Stroudsburg, PA, USA, 2011. ACL.

[13] E. Marsh and D. Perzanowski. Muc-7 evaluation of ie technology: Overview of results. MUC-7, April 1998.

[14] D. Milne and I. H. Witten. An open-source toolkit for mining wikipedia. *Artif. Intell.*, 194:222–239, 2013.

[15] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. EMNLP '09, pages 248–256, Stroudsburg, PA, USA, 2009. ACL.

[17] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. CoNLL '09, pages 147–155. ACL, 2009.

[18] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. HLT '11, pages 1375–1384. ACL, 2011.

[19] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. ACL.

[20] S. Saha and A. Ekbal. Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition. *Data Knowl. Eng.*, 85:15–39, May 2013.

[21] L. Si, T. Kanungo, and X. Huang. Boosting performance of bio-entity recognition by combining results from multiple systems. BIOKDD '05, pages 76–83, New York, NY, USA, 2005. ACM.

[22] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. CONLL '03, pages 142–147, Stroudsburg, PA, USA, 2003. ACL.

[23] L. Todorovski and S. Džeroski. Combining classifiers with meta decision trees. *Machine Learning*, 50(3):223–249, 2003.

[24] Štefan Dlugolinský, P. Krammer, M. Ciglan, and M. Laclavík. MSM2013 IE Challenge: Annotowatch. In *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*, pages 21–26, 2013.