Combining Named Entity Recognition Methods for Concept Extraction

Štefan Dlugolinský[*]
Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9, 845 07 Bratislava, Slovakia
stefan.dlugolinsky@savba.sk

Abstract

Named entity recognition (NER) is a key task of mining semantics from text. Recent growth of social media raised new challenges for NER. Our evaluation of a number of popular NE recognizers over a micro-post dataset showed a significant drop-off in results quality. Current state-of-the-art NER methods perform much better on formal text than on micro-posts. However, the experiment provided us with an interesting observation – although individual NER tools did not perform very well on micro-post data, we have received recall over 91% when we merged all the results of the examined tools. This means that if we were able to combine different NE recognizers in a meaningful way, we might be able to get NER in micro-posts of a very high quality. We propose a method for NER in micro-posts, 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 machine learning techniques, namely decision tree and random forest using the C4.5 algorithm. Evaluation on a standard dataset shows that the proposed approach outperforms underlying NER methods as well as the state-of-the-art NE recognizer specially trained on the micro-post data. 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

I.2 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing—*Language parsing and understanding, Text analysis*

Keywords

named entity recognition, machine learning, micro-posts

1. Introduction

Recent years have seen a significant growth in social media interaction. People are able to interact via the Internet from almost anywhere at any time. They can share their experiences, thoughts and knowledge instantly and they do so in mass dimensions. The easiest and probably the most popular way of interaction on the Web is through micro-posts – short text messages posted on the Web. There numerous such services offering such communication. Notorious examples of micro-posts include tweets, Facebook statuses, comments, Google+ posts, Instagram photos. Micro-posts analysis has a big potential in hidden knowledge that can be used in wide range of domains such as emergency response, public opinion assessment, business or political sentiment analysis and many more. The most important task in order to analyse and make sense of micro-posts is the Named Entity Recognition (NER). NER in micro-posts is a challenging problem due to the limited size of a single micro-post, prevalence of term ambiguity, noisy content, multilingualism [3]. These are the main reasons why existing NER methods perform better on formal newswire text than on micro-posts and there is clearly a space for new methods of NER designed for social media streams. In a thesis, introduced by the current paper, 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 micro-posts. 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 [17] that was used to train decision tree (DT) and random forest (RF) models. The resulting classification model outperformed the best of the combined recognizers by gaining more than 16% in F_1 score and the best baseline model by gaining 1% in F_1 score.

The main contributions of the work are following:

- We show that although existing NER tools designed for news text do not perform well on micro-posts, by merging results of several different NER tools,

[*]Recommended by thesis supervisor:
Assoc. Prof. Dr. Michal Laclavík
Defended at Institute of Informatics, Slovak Academy of Sciences on TODO: MONTH DD, YYYY.

we can achieve high recall and precision.

- 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.

- We provide an extensive evaluation of popular classification models to assess their suitability for the problem of combining results of NER tools. For the best performing ones, we study the influence of algorithms parameters on the classification results.

The paper is structured as follows: Section 2 presents a state-of-the-art related to NER in micro-posts through combining multiple NER methods. Section 3 briefly describes several popular NER tools, which are evaluated over a standard micro-posts dataset in Section 4. Results show a dramatic drop in quality measures compared to the numbers reported on news datasets. Section 5defines baseline NE recognizers, and explains our approach of combining NER tools and evaluates our NE recognition models. Finally, Section 6 summarizes our results and concludes the paper.

2. Present State of the Art

This section presents a state-of-the-art related to NER in micro-posts, which is based on a combination of multiple methods. Regarding the NER for tweets, a similar approach has been taken by Liu et al. [15]. 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. [9] address 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, ±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[1]. The experiments show that the language independent features lead to F_1 score increase and the model outperforms Ritter et al. [22]. Ritter et al. [22] 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 Labelled LDA [18] to exploit Freebase[2] dictionaries as a source of distant supervision. TwiNER, a novel unsupervised NER system for targeted tweet streams is proposed by Li et al. [14]. Similarly to Etter et al. [9], TwiNER does not rely on any linguistic features of the text. It aggregates information from the Web and Wikipedia. The advantage of TwiNER is that it does not require manually annotated training sets. Alternatively, 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 [12], 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 labelled tweet data. The CRF involves POS tags and capitalization of the words as features. The SVM segments tweet using Li et al. [14] 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. NE Recognizers Considered for Combining

We chose various existing NER recognizers based on state-of-the-art NER methods as candidates for combining in classification models discussed later. The list of these tools was complemented by Miscinator, our NE recognizer specially designed for Making Sense of Microposts 2013 (#MSM2013) Concept Extraction Challenge. Below, we briefly describe the recognizers focusing on which methods they use for NER and how they were configured.

ANNIE [6] relies on finite state algorithms, gazetteers and the JAPE (Java Annotation Patterns Engine) language [7]. We have used ANNIE from GATE Developer 7.1.

Apache OpenNLP[3] is based on maximum entropy models [21] and perceptron learning algorithm [23]. We have used Apache OpenNLP v1.5.2.

Illinois Named Entity Tagger [19] uses a regularized averaged perceptron [11] with external knowledge (un-labelled text, gazetteers built from Wikipedia and word class models). We have used Illinois NET v1.0.4 with 4-label type set and default configuration.

Illinois Wikifier [20] is based on a Ranking SVM (Support Vector Machine) [5] and exploits Wikipedia link structure in disambiguation. We have used Illinois Wikifier 1.0[4].

Open Calais operates behind a shroud of mystery since there is not much information available about how its NE recognition works. Official sources[5] say, that it uses NLP, machine learning and other methods as well as Linked data.

Stanford Named Entity Recognizer [10] is based on CRF (Conditional Random Field) sequence models [13]. We have used Stanford NER v1.2.7[6] with English 4-class caseless CONLL model[7].

Wikipedia Miner[8] **[16]** is a text annotation tool, which is capable of annotating Wikipedia topics in a given text. It exploits Wikipedia link graph and Wikipedia category hierarchy and relies on machine learning classifiers, which are used for measuring relatedness of concepts and terms, as well as for measuring disambiguation. We have used this software

[1] There were about 40,000 tweets tagged in more than two weeks by Amazon Mechanical Turk

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

[4] `http://cogcomp.cs.illinois.edu/page/download_view/Wikifier`

[5] `http://www.opencalais.com/about`

[6] `http://nlp.stanford.edu/software/stanford-ner-2012-11-11.zip`

[7] english.conll.4class.caseless.distsim.crf.ser.gz

to discover Wikipedia topics in micro-posts. Discovered topics were then tagged according to DBPedia Ontology[9].

Miscinator was our gazetteer tool specially designed for annotating MISC entities in the #MSM2013 Concept Extraction Challenge. The gazetteer was constructed by extending the MISC annotations from the training set with Google Sets[10].

Most of the NE recognizers were based on statistical learning methods. Some of them used also gazetteers and other external knowledge such as Wikipedia or Linked Data. Outline of the NE recognizers is shown in Figure 1.

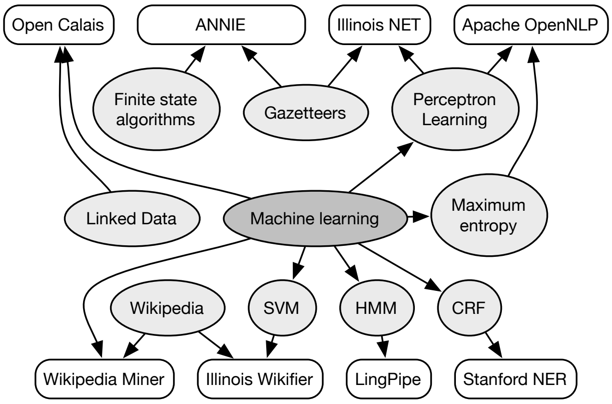


Figure 1: Outline of NE recognizers

4. Evaluation of NE Recognizers

In this section, we provide an evaluation of the NER recognizers described in Section 3 over micro-post data. Our intent was to observe the performance of each individual NE recognizer before combining it with other NER tools. The evaluation was also focused on analysis, which NE recognizer is more suitable for particular NE class and whether NE recognizers produce diverse results. Evaluated NE recognizers were not specially configured, tweaked or trained for micro-posts prior to the evaluation. They used their default configuration for formal English text. The reason for this was that we wanted to see how they cope with a different kind of text that they were trained for. As they differed in supported NE classes, it was necessary to align them with our taxonomy (LOC, MISC, ORG, PER). This was achieved by simple mapping; e.g. Person→PER. If the mapping was not possible, we skipped the particular NE class and we did not include it in computation of evaluation metrics; e.g., skipped MISC class for ANNIE, Apache OpenNLP and LingPipe.

4.1 Dataset

NE recognizers were evaluated over the adapted #MSM2013 IE Challenge training dataset [2]. We took the 1.5 version and cleaned it from duplicate micro-posts as well as from micro-posts overlapping the test dataset. The cleaned training dataset finally contained 2752 unique manually annotated micro-posts with classification restricted to four entity types:

PER – full or partial person names

LOC – full or partial (geographical or physical) location names, including: cities, provinces or states, countries, continents and (physical) facilities

ORG – full or partial organization names, including academic, state, governmental, military and business or enterprise organizations

MISC – a concept not covered by any of the categories above, yet limited to one of the entity types: film/movie, entertainment award event, political event, programming language, sporting event and TV show.

We also adapted the test dataset from the #MSM2013 IE Challenge over 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 work are also available for download[11] in GATE SerialDataStore format. Datasets include results of all the used NE recognizers as well as our NER models discussed later in this chapter.

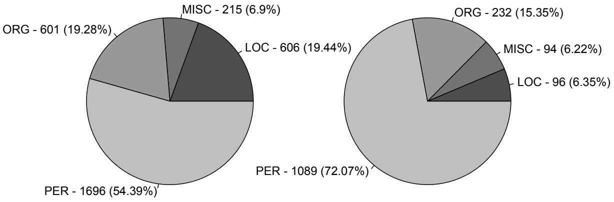


Figure 2: Occurence of named entities in train (left) and test (right) datasets

4.2 Evaluation Results

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. It turned out that the best performing NE tagger on the evaluation dataset was OpenCalais, which was the best in recognizing LOC and ORG entities. The second was Illinois NET, which was the best in recognizing PER entities. The best tool in recognizing MISC entities was Miscinator, which achieved 48% in F_1 score. Further details about the evaluation can be found in [8]. Some of the evaluation results may slightly differ from those displayed in Table 1. It is for the reason that we accepted adjectival and demonymic forms for countries as *MISC* type; e.g., Slovak (adjectival), Slovaks (demonym).

4.3 Theoretical Gain in Performance

The current evaluation also shows that the NE recognizers produced diverse annotations. This behavior can be seen in increased recall after the results were unified from all of the taggers and cleaned from duplicates. Figure 3 and Figure 4 illustrate the situation and possible recall, which can theoretically be achieved when combining the recognizers. Gain in recall on a per NE class basis is computed

[9] `http://dbpedia.org/Ontology`

[10] `http://googlesystem.blogspot.com/2012/11/google-sets-still-available.html`

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

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
Open Calais	**0.737**	0.270	**0.557**	0.692	0.657	0.503	**0.564**	**0.720**	0.598	**0.653**
Illinois NET	0.721	0.105	0.359	**0.789**	0.488	0.505	0.493	0.607	**0.645**	0.626
Stanford NER	0.670	0.053	0.292	0.747	0.446	0.436	0.440	0.597	0.591	0.594
ANNIE	0.677	–	0.356	0.606	0.711	0.369	0.410	0.636	0.483	0.549
Illinois Wikifier	0.552	0.159	0.509	0.624	0.541	0.419	0.461	0.624	0.472	0.537
Apache OpenNLP	0.507	–	0.270	0.579	0.679	0.281	0.339	0.624	0.384	0.475
Wikipedia Miner	0.560	0.062	0.327	0.613	0.346	**0.524**	0.391	0.321	0.573	0.412
LingPipe	0.349	–	0.071	0.348	0.400	0.300	0.192	0.161	0.381	0.226
Miscinator	–	**0.479**	–	–	**0.922**	0.092	0.120	0.687	0.025	0.049

in Table 2 together with overall recall using the macro and micro averaging. Gain values were calculated with respect to score of the best tool for particular NE class. Macro and Micro values were calculated analogously. If the taggers would not produce diverse results, the recall of unified and de-duplicated results will copy the performance of the tools. Therefore, we see a place for theoretical gain in performance of NER when combining the tools together. Yet the increased recall of unified taggers comes hand in hand with lower precision. This is because the number of false positive (FP) tags can grow with every new tagger added to the union. We expect that there can be a combining model trained, which would decrease the number of false positive (FP) entities produced by combined taggers and thus increase the precision, while keeping the number of true positives (TP) and therefore recall still relatively high.

Nevertheless, the real gain in recall can be higher than the theoretical gain calculated in Table 2. This is possible if combining model is capable of decreasing the number of false negatives (FN) occurring for combined taggers; i.e., decreasing the number of missed entities. The number of false negatives (FN) can be decreased if combining model counts on true negative (TN) entities produced by combined taggers and transforms them to true positive (TP) entities or false positive (FP) entities. If they are transformed to true positives (TP) then recall together with precision are increased. If they are transformed to false positives (FP) then only recall is increased and precision is decreased. Of course, combining model should not produce less true positive (TP) entities than the combined taggers together to gain the recall. True negative (TN) entities, recognized by combined taggers, can be properly chosen NEs having classification out of the target taxonomy; e.g. NP – noun phrase. If such NP entity is taken and transformed to a true positive (TP) entity of the target taxonomy, for instance PER, then recall is increased as the number of true positives (TP) is increased too. Moreover, precision is also increased. Therefore, we chose true negatives (TN) to be involved in training of combination model described later in Section 5.

Table 2: Gain in recall that could be theoretically achieved if combining taggers together

	Theoretical gain in recall [%]					
dataset	LOC	MISC	ORG	PER	Mac.	Mic.
train	26.4	96.2	73.7	22.7	63.3	40.7
test	43.1	90.5	91.7	16.0	64.2	29.0

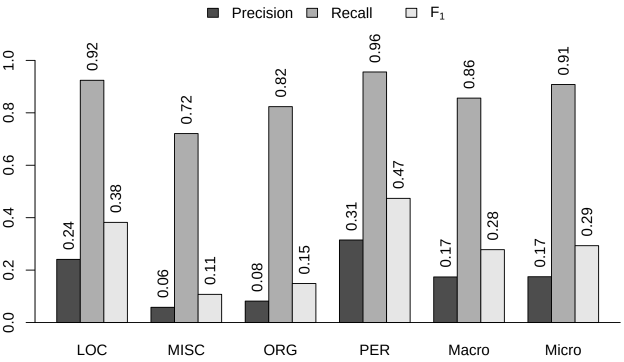


Figure 3: Precision, Recall and F_1 of unified NE recognizers over the train dataset

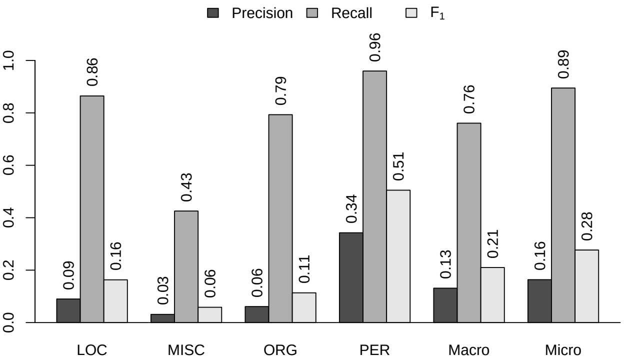


Figure 4: Precision, Recall and F_1 of unified NE recognizers over the test dataset

Results of the evaluation and the experiment with merged annotations showed that there was a place for theoretical gain in overall performance of NER if the taggers were combined together.

5. Combining NE Recognizers

The idea of how to combine NE recognizers was to use machine learning techniques to build a classification model, which would be trained on features describing microposts' text as well as annotations produced by involved NE recognizers. We used the training dataset to build the model and the test dataset for evaluating it and comparing with other NE recognizers (Section 4).

According to the evaluation results in section 4, we chose for combining seven out of eight NE recognizers based on different methods. The discarded one was LingPipe because its model *English News: MUC-6* was not suitable for micro-post texts, despite it fitted as the best for this task from all of the tree available LingPipe's models. The other two available LingPipe models were English Genes: GeneTag and English Genomics: GENIA. Seven chosen NE recognizers were then complemented by Miscinator tool.

As the 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 machine learning techniques, but it was necessary to transform this problem into a standard machine learning task. In this case it was suitable to transform the task of NER into a task of classification. The intent was that machine learning 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 does not belong to any of the four target classes. Then a simple algorithm can be applied to merge the re-classified annotations into the final results.

5.1 Baseline NE Recognizers

There were three baseline NE recognizers defined, which we used to compare the performance of the combining models.

The first baseline, Baseline Train, was defined as a straightforward composition of the best NE recognizers in each NE class according to the evaluation made over the training dataset (Section 4, Table 1); i.e., LOC, MISC and ORG classes were extracted by OpenCalais and PER class was extracted by Illinois NET.

Theoretically, there could be a better baseline assembled if we chose the best NE recognizers according to evaluation on the test dataset instead of the training dataset. It is because there is no guarantee that all the best taggers on one dataset are the best on a different one. Therefore, we defined a theoretical baseline, Baseline Test, which is assembled from the best taggers according to evaluation made over the test dataset. This baseline is theoretical, because in real life we do not know in advance which taggers for which NE class should we assemble.

The third, Baseline SNER, was a model based on Stanford NER CRFClassifier trained on the training dataset. To train this model, we adapted properties of the out-of-the-box model english.conll.4class.caseless.distsim.crf.ser.gz[12].

The performance of the baselines can be seen in Table 3 together with performances of the NE recognizers considered for combining. The evaluation was made over the test dataset. Results are ordered by Micro avg. F_1 score. As was expected, the baselines outperformed underlying NE recognizers in precision and F_1 measures. Our goal was to overcome the performance of the baselines with a combining model produced by machine learning approach.

5.2 Transforming NE Annotations into Vectors

We took an approach of describing how particular methods performed on different entity types compared to the response of other methods and 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 5).

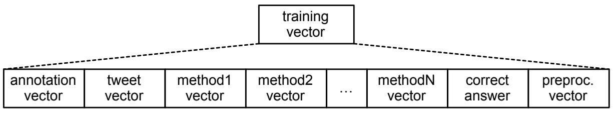


Figure 5: Training vector

The first sub-vector of the training vector was an annotation vector (Figure 6). 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.

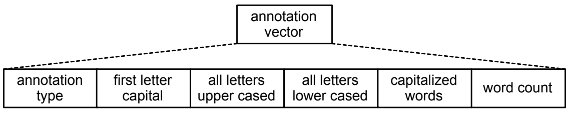


Figure 6: Annotation vector

The second sub-vector described micro-posts as a whole (Figure 7). It contained features describing whether all words longer than four characters were capitalized, uppercase, or lowercase. We called this sub-vector tweet vector.

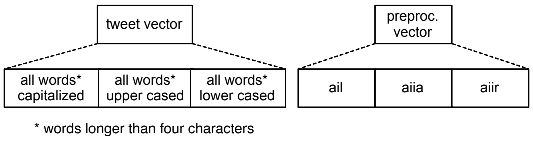


Figure 7: Tweet vector (left) and preprocessing vector (right)

The remainder 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 8). 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.

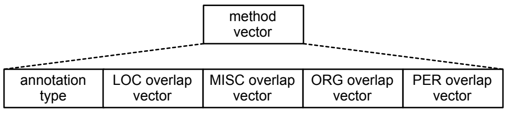


Figure 8: Method vector

[12] https://github.com/stanfordnlp/CoreNLP/raw/fe2a9672bd7beb589d245d13d20e89754c06917f/scripts/ner/english.conll.4class.caseless.distsim.prop

Table 3: 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 SNER	0.589	0.267	**0.465**	**0.864**	0.689	0.484	0.546	**0.836**	**0.710**	**0.768**
Baseline Test	**0.614**	**0.295**	0.464	0.844	0.669	**0.491**	**0.554**	0.801	0.706	0.750
Baseline Train	**0.614**	**0.295**	0.296	0.844	0.694	0.436	0.512	0.831	0.672	0.743
Stanford NER	0.513	0.000	0.302	0.822	0.393	0.431	0.409	0.673	0.668	0.670
Illinois NET	0.500	0.058	0.317	0.844	0.407	0.462	0.430	0.647	0.694	0.669
Open Calais	**0.614**	**0.295**	0.296	0.691	0.643	0.412	0.474	0.656	0.602	0.628
ANNIE	0.480	–	0.194	0.679	0.605	0.325	0.338	0.633	0.519	0.570
Illinois Wikifier	0.343	0.087	0.464	0.677	0.439	0.382	0.393	0.628	0.496	0.554
Apache OpenNLP	0.384	–	0.126	0.637	0.571	0.265	0.287	0.619	0.430	0.508
Wikipedia Miner	0.292	0.039	0.288	0.671	0.287	0.463	0.322	0.323	0.571	0.413
LingPipe	0.147	–	0.046	0.385	0.364	0.277	0.145	0.147	0.378	0.211
Miscinator	–	0.191	–	–	**0.881**	0.029	0.048	0.524	0.007	0.014

Each overlap vector of a particular method and NE class (Figure 9) consisted of five components – *ail*: the average intersection length of a reference annotation with the method 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*.

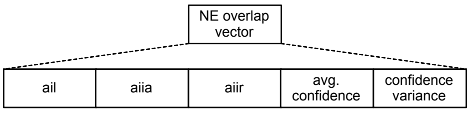


Figure 9: Overlap vector

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.

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

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.

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

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.

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

A simple example of overlap vector computation is shown in Figure 10. 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 remainder 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, due to the eight annotations taken as reference annotations, where also the manual annotation PER is included.

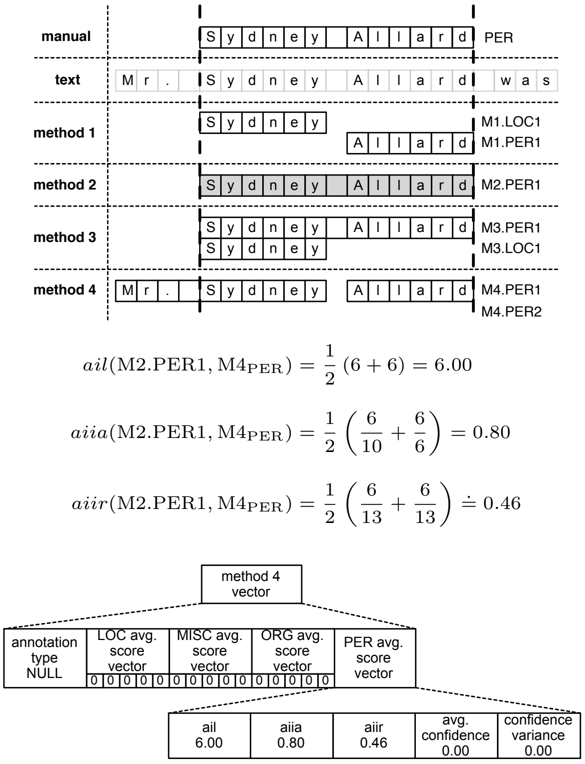
The final two components in the training vector were the correct answer (i.e., the correct annotation type taken from manual annotation) and a special preprocessing vector (Figure 7). The pre-processing vector included three components: *ail*, *aiia* and *aiir*, which described the intersection of the reference annotation when it was correct with the correct answer. If the reference annotation was not correct the values of the pre-processing 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.

5.3 Training Data Preprocessing

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

According to the pre-processing vector (Figure 7), there were training vectors removed, in which the *annotation*



$$ail(\mathrm{M2.PER1}, \mathrm{M4}_{\mathrm{PER}}) = \frac{1}{2}(6+6) = 6.00$$

$$aiia(\mathrm{M2.PER1}, \mathrm{M4}_{\mathrm{PER}}) = \frac{1}{2}\left(\frac{6}{10} + \frac{6}{6}\right) = 0.80$$

$$aiir(\mathrm{M2.PER1}, \mathrm{M4}_{\mathrm{PER}}) = \frac{1}{2}\left(\frac{6}{13} + \frac{6}{13}\right) \doteq 0.46$$

Figure 10: Example of overlap vector computation

type attribute in the *annotation vector* was correct but the *aiir* attribute in the pre-processing 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 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 recognizers 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 *pre-processing vector* have been also removed.

The pre-processing phase, described above, 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 ∼ 63K training vectors with ∼ 200 attributes and finished on ∼ 31K unique records with ∼ 100 highly relevant attributes.

5.4 Model Training and Evaluation

We tried several algorithms to train different classification model candidates, which we compared each other according to F$_1$ score. We also examined AUROC (The Area Under an ROC curve - Receiver Operating Characteristic curve) 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, as it avoided overfitting effect without a need of another test dataset. The best performance was achieved by DT classification model built with J48[13] algorithm (DTJ48) followed by RF [4] model. The third was a classification model based on REPTree (Reduced Error Pruned Tree) built with Bagging algorithm (Table 4).

Table 4: 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

We focused on the first two best performing algorithms and built several classification models while varying some of the 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 decision trees and *number of trees* for random forest. The classification models were evaluated using a hold-out validation method over the test dataset. Evaluation results are displayed in Table 6. The best performing models were based on random forest, namely RF N100, RF N200, RF N300 and RF N400. These models outperformed models based on decision trees as well as baseline recognizers and all the combined NE recognizers. We can see that recall and precision were growing with the number of trees for the random forest models and continued to converge to 79% and 76% respectively. This behavior is more evident in Figure 11, where F$_1$ measures are shown for particular NE classes according to the variated number of trees. Dashed lines indicate score of the baselines, i.e., Baseline SNER, Baseline Test and Baseline Train. The performance of the Test and Train baselines was the same for LOC, MISC and PER classes since they used different tagger only for ORG class (see section 5.1 for more details). Therefore, their lines interlap each other in the graph for LOC, MISC and PER classes.

Evaluation results of the models built with J48 algorithm (C4.5 implementation) while varying the M parameter are displayed in Figure 12. Although these models did not outperform the best baseline, one of the models, DTJ48 M13, was slightly better than the rest of the baseline models.

The #MSM2013 21_3 model in the Table 6 was our submission to the #MSM2013 IE Challenge [24]. This model was one of our early models and finished in the challenge as the first runner-up with a loss of 1% in F$_1$ on the winner Habib et. al [12]. #MSM2013 21_3 model was the second best in precision and the best in recall in the challenge. Results of this model in the table may be slightly worse

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

Table 5: Mean and standard deviation of the combining models evaluated over the test dataset

		F_1				Macro avg.			Micro avg.		
Models		LOC	MISC	ORG	PER	P	R	F_1	P	R	F_1
RF	μ	0.560	0.233	0.450	0.868	0.548	0.527	0.528	0.751	0.752	0.751
	σ	0.032	0.031	0.049	0.014	0.050	0.017	0.029	0.044	0.015	0.030
DTJ48	μ	0.543	0.281	0.394	0.857	0.547	0.508	0.519	0.754	0.723	0.738
	σ	0.048	0.048	0.024	0.011	0.033	0.009	0.016	0.023	0.006	0.014

Table 6: Evaluation of NER 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 N100	0.584	0.231	0.476	**0.883**	0.589	0.529	0.543	0.788	0.760	**0.774**
RF N200	0.593	0.236	0.484	0.878	0.604	0.529	0.548	0.789	0.759	0.774
RF N300	0.600	0.234	**0.491**	0.876	0.602	0.533	0.550	0.788	0.758	0.773
RF N400	0.597	0.234	0.490	0.876	0.601	0.533	0.549	0.788	0.758	0.772
Baseline SNER	0.589	0.267	0.465	0.864	0.689	0.484	0.546	**0.836**	0.710	0.768
RF N17	0.557	0.262	0.476	0.876	0.559	0.542	0.543	0.766	**0.762**	0.764
RF N21	0.554	0.257	0.474	0.874	0.562	0.537	0.540	0.766	0.758	0.762
RF N11	0.570	0.247	0.463	0.873	0.554	0.535	0.538	0.763	0.758	0.761
RF N14	0.550	0.259	0.456	0.877	0.548	0.536	0.535	0.761	0.760	0.760
RF N9	0.569	0.255	0.475	0.867	0.551	**0.544**	0.542	0.755	0.760	0.758
RF N7	0.562	0.252	0.444	0.868	0.537	0.537	0.531	0.746	0.758	0.752
DTJ48 M13	0.570	**0.356**	0.365	0.867	0.599	0.516	0.539	0.775	0.729	0.751
Baseline Test	**0.614**	0.295	0.464	0.844	0.669	0.491	**0.554**	0.801	0.706	0.750
DTJ48 M11	0.585	0.268	0.400	0.863	0.560	0.515	0.529	0.766	0.726	0.746
DTJ48 M9	0.549	0.288	0.388	0.864	0.554	0.510	0.522	0.770	0.723	0.745
Baseline Train	**0.614**	0.295	0.296	0.844	**0.694**	0.436	0.512	0.831	0.672	0.743
DTJ48 M7	0.567	0.231	0.411	0.858	0.535	0.510	0.517	0.754	0.726	0.740
RF N5	0.530	0.221	0.420	0.859	0.511	0.518	0.507	0.727	0.747	0.737
DTJ48 M5	0.536	0.232	0.427	0.854	0.526	0.507	0.512	0.750	0.722	0.736
#MSM2013 21_3	0.505	0.308	0.411	0.834	0.510	0.532	0.514	0.701	0.726	0.713
DTJ48 M2	0.453	0.312	0.372	0.836	0.504	0.492	0.493	0.711	0.712	0.712
RF N3	0.500	0.195	0.368	0.846	0.466	0.496	0.477	0.685	0.730	0.707
RF N2	0.508	0.151	0.333	0.836	0.445	0.489	0.457	0.643	0.711	0.675

than the official challenge results[14], 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 comparison we present precision, recall and F_1 measures of the best performing model – RF N100, best DT model – DTJ48 M13, baseline recognizers and the top three combined NE recognizers in Figure 13. The highest score in precision was achieved by Baseline SNER followed by the rest of the baselines. RF N100 and DTJ48 M13 were fourth and fifth respectively. However, they performed better as any of the combined NE recognizers. RF N100 gained 17% and DTJ48 M13 15% in precision with respect to Stanford NER as the best in precision among the combined NE recognizers. The loss of RF N100 on Baseline SNER was 6%. The highest score in recall was achieved by RF N100 followed by DTJ48 M13. The gain in recall of the RF N100 model was 7% with respect to the best baseline – Baseline SNER and 10% with respect to the best combined NE recognizer – Illinois NET. The highest score in F_1 measure was achieved by RF N100. The gain in F_1 of the RF N100 model was 1% with respect to the best baseline – Baseline SNER and 16% with respect to the best combined NE recognizer – Stanford NER. DTJ48 M13 was the third with 2% loss in F_1 on the second Baseline SNER, but with 12% gain in F_1 with respect to Stanford NER.

The gain in F_1 scores was a sign that combining models were capable of eliminating false positive (FP) entities and/or transforming true negative (TN) entities into true positive (TP) entities. To confirm this assumption, we compared the results of the best combining model, RF N100, with merged and de-duplicated results of the combined tools. Results of the alignments are in Table 7. Although we did not notice that there were any true negative (TN) entities transformed to true positive (TP) entities, but instead there were 194 true positive (TP) entities transformed to false negative (FN) entities, we noticed that there were 3 487 false positive (FP) entities eliminated. This caused the RF N100 model to gain 199.5% in precision with respect to the merged and de-duplicated results of the combined tools. The decrease of true positive (TP) entities and the increase of false negative (FN) entities led to 14.45% loss in recall, but the 33.08% gain in F_1 was still relatively high and showed that there can be combining models trained with superior performance

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

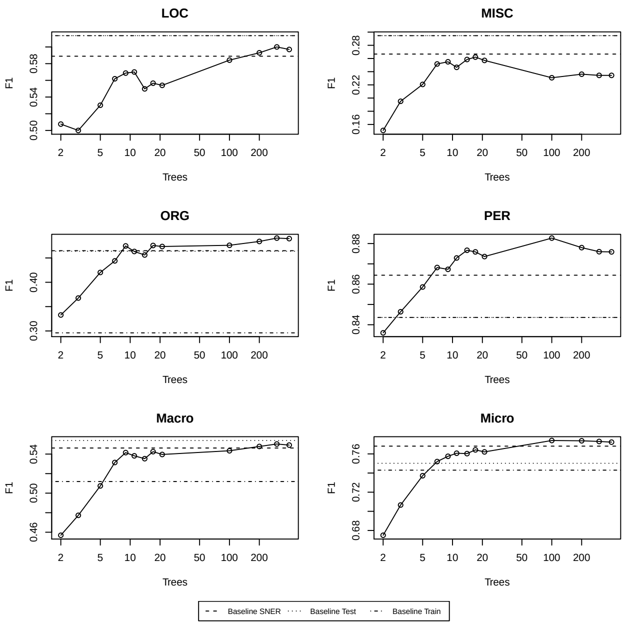


Figure 11: Impact on F$_1$ while varying number of trees for Random Forest algorithm

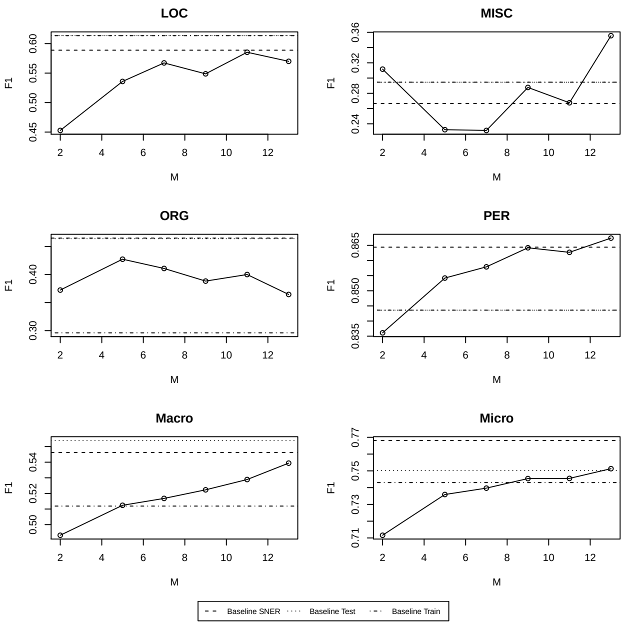


Figure 12: Impact on F$_1$ while varying parameter M for Decision Tree J48 (C4.5) algorithm

to that of the combined tools.

Table 7: Performance analysis of the RF N100 model over the #MSM2013 Test dataset

		TP		FN	FP
NE	Model	COR	PAR	MIS	SPU
LOC	merged+dedup.	82	8	6	425
	RF N100	59	7	30	40
	Δ	−23	−1	24	−385
MISC	merged+dedup.	40	12	42	1 224
	RF N100	15	7	72	14
	Δ	−25	−5	30	−1 210
ORG	merged+dedup.	179	18	35	921
	RF N100	104	35	93	66
	Δ	−75	17	58	−855
PER	merged+dedup.	1 042	20	27	1 133
	RF N100	971	44	74	96
	Δ	−71	24	47	−1 037
All	merged+dedup.	1 343	58	110	3 703
	RF N100	1 149	93	269	216
	Δ	−194	35	159	−3 487

A closer analysis of the annotation results indicates that there were many results correctly classified, but such results did not exactly match the 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 also removed definite articles from LOC and PER results. Moreover, we removed titles from PER results; e.g., Dr., Mr. or Sir. Evaluation of models with this simple post-processing (PP) is displayed in Table 8. We applied post-processing on the best performing RF and DT models as well as on the best baseline – Baseline SNER, where the gain in F$_1$ with respect to models without post-processing was 1.9%, 2.8% and 0.3% respectively. The highest score in F$_1$ measure was achieved by RF N100 PP model, which gained 2.5% over the best baseline – Baseline SNER PP.

6. Conclusions

We introduced an approach to combine NE recognizers based on diverse methods on a task of NER in microposts and examined several machine learning techniques for the combination of text and annotation features produced by the recognizers. The best performing machine learning techniques were random forest and decision trees based on the C4.5 algorithm. Combining models built on top of these techniques achieved performance superior to those of the combined NE recognizers. Moreover, the best combining model, RF N100, trained over the informal text of micro-posts performed better than the baseline recognizers, although the combined NE recognizers were not specially trained or tweaked on infomal text. The gain in F$_1$ of the RF N100 model with respect to the best of the combined NE recognizers, Stanford NER, was 16% and 1% with respect to the best baseline recognizer, which was also Stanford NER, but specially trained on the micro-posts data. Performance of the RF and DT models indicates that machine learning techniques lead to more favorable combination of underlying NE recognizers than was conducted manually in one of the baseline NE recognizers, which was an ensemble of the best NE recognizers for each NE class. The gain in F$_1$ of the

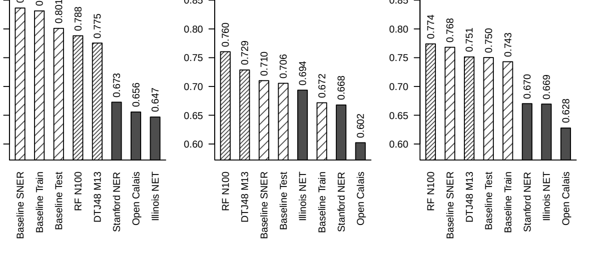


Figure 13: Comparison of our two best performing models RF N100 and DTJ48 M13 with the baselines and top-three combined NE recognizers

Table 8: 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
RF N100 PP	0.594	0.246	**0.544**	**0.887**	0.618	**0.550**	**0.568**	0.804	**0.774**	**0.789**
RF N100	0.584	0.231	0.476	0.883	0.589	0.529	0.543	0.788	0.760	0.774
DTJ48 M13 PP	0.576	**0.356**	0.441	0.880	0.626	0.537	0.563	0.796	0.750	0.772
Baseline SNER PP	0.578	0.267	0.465	0.868	0.687	0.482	0.544	**0.839**	0.712	0.770
Baseline SNER	0.589	0.267	0.465	0.864	0.689	0.484	0.546	0.836	0.710	0.768
DTJ48 M13	0.570	**0.356**	0.365	0.867	0.599	0.516	0.539	0.775	0.729	0.751
Baseline Test	**0.614**	0.295	0.464	0.844	0.669	0.491	0.554	0.801	0.706	0.750
Baseline Train	**0.614**	0.295	0.296	0.844	**0.694**	0.436	0.512	0.831	0.672	0.743
#MSM2013 21_3	0.505	0.308	0.411	0.834	0.510	0.532	0.514	0.701	0.726	0.713

RF N100 model with respect to the ensemble baseline was approx. 3%. The advantage of the combining models is that they can adapt to actual text according to its features and annotations from combined NE recognizers, as well as benefit from given negative examples as we saw it in their capability to eliminate false positive (FP) results given by combined tools. The proposed approach of combining NER methods was successfully applied in the #MSM2013 Concept Extraction Challenge organized under WWW2013 and finished as the first runner-up with $F_1 = 66.2\,\%$ and a 1.2 % loss. More specifically, we were the first in recall score ($R = 61.3\,\%$) and the second in precision score ($P = 76.4\,\%$).

Acknowledgements. The author would like to thank his supervisor Assoc. Prof. Dr. Michal Laclavík for his valuable advice and comments.

References

[1] A. E. C. Basave, M. Rowe, M. Stankovic, and A.-S. Dadzie, editors. *Proceedings, Concept Extraction Challenge at the 3rd Workshop on Making Sense of Microposts (#MSM2013): Big things come in small packages, Rio de Janeiro, Brazil, 13 May 2013*, May 2013.

[2] A. E. C. Basave, A. Varga, M. Rowe, M. Stankovic, and A.-S. Dadzie. Making sense of microposts (#msm2013) concept extraction challenge. In Basave et al. [1], pages 1–15.

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

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

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[7] H. Cunningham, D. Maynard, and V. Tablan. Jape: a java annotation patterns engine. Technical Report CS-00-10, University of Sheffield, UK, November 2000.

[8] S. Dlugolinsky, M. Ciglan, and M. Laclavik. Evaluation of named entity recognition tools on microposts. In *Intelligent Engineering Systems (INES), 2013 IEEE 17th International Conference on*, 2013.

[9] D. Etter, F. Ferraro, R. Cotterell, O. Buzek, and B. Van Durme. Nerit: Named entity recognition for informal text. Technical report, Technical Report 11, Human Language Technology Center of Excellence, Johns Hopkins University, July, 2013.

[10] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[11] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[12] M. Habib, M. V. Keulen, and Z. Zhu. Concept extraction challenge: University of Twente at #msm2013. In Basave et al. [1], pages 17–20.

[13] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[14] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee. Twiner: Named entity recognition in targeted twitter stream. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 721–730, New York, NY, USA, 2012. ACM.

[15] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 359–367, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

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

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

[18] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 248–256, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[19] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[20] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[21] A. Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution*. PhD thesis, University of Pennsylvania, 1998.

[22] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[23] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[24] Štefan Dlugolinský, P. Krammer, M. Ciglan, and M. Laclavík. MSM2013 IE Challenge: Annotowatch. In Basave et al. [1], pages 21–26.

Selected Papers by the Author

S. Dlugolinsky, M. Laclavik, L. Hluchy. Towards a search system for the web exploiting spatial data of a web document. in *Database and Expert Systems Applications (DEXA), 2010 Workshop on*, Aug 2010, pp. 27–31.

S. Dlugolinsky, M. Seleng, M. Laclavik, L. Hluchy. Distributed web-scale infrastructure for crawling, indexing and search with semantic support. *Computer Science*, vol. 13, no. 4, 2012. [Online]. Available: `http://journals.agh.edu.pl/csci/article/view/42`

Š. Dlugolinský, P. Krammer, M. Ciglan, M. Laclavík. MSM2013 IE Challenge: Annotowatch. in *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*, A. E. C. Basave, M. Rowe, M. Stankovic, and A.-S. Dadzie, Eds., May 2013, pp. 21–26. [Online]. Available: `http://ceur-ws.org/Vol-1019/paper_21.pdf`

Š. Dlugolinský, M. Laclavík, M. Šeleng, M. Ciglan, M. Tomašek, L. Hluchý. Advanced email search in small enterprises. In 7th Workshop on Intelligent and Knowledge Oriented Technologies. - Bratislava : Nakladateľstvo STU, 2012, p. 23-26. ISBN 978-80-227-3812-5.

S. Dlugolinsky, M. Ciglan, M. Laclavik. Evaluation of named entity recognition tools on micro-posts. in *Intelligent Engineering Systems (INES), 2013 IEEE 17th Int. Conf. on*, 2013.

Š. Dlugolinský, T. G. Nguyen, M. Laclavík, M. Šeleng. Character gazetteer for named entity recognition with linear matching complexity. In Proceedings of the 2013 World Congress on Information and Communication Technologies : WICT 2013. Eds. Ngo, L.T. et al. - IEEE Systems Man and Cybernetics Society, Spain Chapter, 2013, p. 364-368. ISBN 978-1-4799-3230-6.

Š. Dlugolinský, P. Krammer, M. Ciglan, M. Laclavík, L. Hluchý. Combining named entity recognition methods for concept extraction in Microposts. In M. Rowe, M. Stankovic, and A.-S. Dadzie, editors, 4th Workshop on Making Sense of Microposts (#Microposts2014), pages 34–41, April 2014.