Entity Recognition Based on the Co-occurrence Graph and Entity Probability

Alan Eckhardt Juraj Hreško Jan Procházka Otakar Smrž
Seznam.cz Research
Radlická 3294/10, 150 00 Prague, Czech Republic
{alan.eckhardt,juraj.hresko,jan.prochazka,otakar.smrz}
@firma.seznam.cz

ABSTRACT

This paper describes our system for Entity Recognition and Disambiguation Challenge 2014. There were two tasks - first one to find entities in queries or and second one to find entities in texts from web pages.

We have participated in both tracks with the same system tuned to each task. The system and its components are described in depth, together with the influence of each of the components on the performance. The specifics of each of the tracks are also discussed.

Categories and Subject Descriptors

I.2.7 [**Computing methodologies**]: Natural language processing—*Information extraction*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Algorithms, Performance

Keywords

entity recognition and disambiguation, entity linking, sense disambiguation

1. INTRODUCTION

One of the peculiar tasks in the area of language understanding is to describe the meaning of a word or a group of words used in some context. This task differs across the word categories from connectives through verbs to substantives. Substantives form a special category alone. If we think about the types of meaning they could bear, we could separate them into two classes. The first one - common nouns - involves expressions which are used to describe some ideals or classes of objects. The second one - proper nouns - are used as pointers to some concrete or abstract entities. Nouns could be considered as the most important objects when building a knowledge base which is a storage of complex information about our world and is accessible by a computer program.

Identifying the proper nouns in the text, as well as their correct meaning (disambiguation), is the key issue which we tried to handle in this work. While the problem with disambiguation of common nouns is heavily dependent on given ontology, it could represent hard problem for a computer as well as for a human being. In case when some pre-given ontology is used, it could be useful mostly for people or domains that are accepting it. Problem with ambiguity is a problem of definition of the meaning.

- Is 'game of chess' a type of 'sport'?
- Can we describe '1992 Los Angeles riots' as a 'war'?

These questions show us the type of ambiguity we can observe while handling common nouns. Do we describe 'sport' as a "competitive physical activity" (Wikipedia[13]) or as "an activity that you do for pleasure and that needs physical effort or skill" (Oxford Advanced American Dictionary)? Is war "an act of violence to compel our opponent to fulfil our will" (Carl von Clausewitz [12]) or a " state-based conflict or dyad ... [with] [a]t least 1000 battle-related deaths in one calendar year." (UCDP [10])?

The proper nouns disambiguation offers a more attainable goal. It could be more or less described as matching one particular (concrete or abstract) object in our world with a part of analysed text. The problem of ontology is a bit less harsh because there is a simple linkage between the name and the object and a well defined domain of objects. The problem with ambiguity is caused by similarity or equality of expressions describing different entities.

- "George Bush faints, then falls after choking on pretzel." (January 2002)
- "George Bush fainted at a banquet hosted by the Prime Minister of Japan." (January 1992)

In the two sentences above we can observe the reference - "George Bush" - which is pointing on two different entities - George W. Bush and George H. W. Bush respectively. But without the context we could only guess if the reference is related to one of the Presidents of the USA, American biblical scholar, young politician (son of Jeb Bush) or former NASCAR driver.

Successful solution of entity recognition can be used for more sophisticated form of data indexing. It covers tagging

ERDC '2014 Gold Coast, Australia

search results, dividing them into groups according to entities from the query with those in the results etc. From users' point of view this could bring more precise outcomes thanks to identifying the right meaning by the search engine or by refining the objects by user selection of the desired meaning.

In this paper we will describe our system for entity detection and disambiguation, which is being developed for internal purposes of the company Seznam.cz. The system was appropriately modified so it follows the rules of Entity Recognition and Disambiguation Challenge. The objective is, as ERDC organizers put it, : " to recognize mentions of entities in a given text, disambiguate them, and map them to the entities in a given entity collection or knowledge base."

The ERD Challenge consists of two different parts, the Short Track and the Long Track. Short Track systems are intended to search for all possible entities in texts of search queries. Long Track systems' goal is to identify particular entities occurring in short articles.

There are few important distinctions between those two tasks. First, queries are non-formatted pieces of texts with possibly wrong grammar and no capital letters used. Second, sometimes the ambiguity of entities in queries could not be resolved sometimes - therefore there can be more than just one solution. Third, the rules specified that for the Long Track it is needed to determine not only the occurrence but the particular position of an entity.

2. RELATED WORK

In this chapter we summarize existing approaches and techniques recently developed for ERD in general. ERD systems are usually divided into three parts:

- mention identification
- collection of entity candidates for each mention
- candidates disambiguation

Usually all three parts depend on each other and are difficult to separate. The mention identification selects relevant parts of the analysed text. The relevant parts of the text are defined by the purpose why we run ERD. Mentions could be based on chosen token classes from Part-of-speech tagging (POS tagging) analysis, named entities or entity names from a knowledgebase, such as Wikipedia or Freebase, and its morphological variants.

Candidates for each mention are collected by looking up entities in a knowledgebase or can be retrieved from a database created by clustering the same mention that occurs in different contexts. The candidates are then ranked or pruned.

Entity disambiguation approaches can be classified into groups depending on disambiguation strategy.

Local level disambiguation

Entities are resolved by exploiting the local context only. In [8], they use other mention candidates to disambiguate their mentions against each other. Relatedness between candidates (Wikipedia articles) and Google Distance inspired measure are computed and averaged at first. The measures take into acount article incoming links for relatedness resp. outgoing links for Google Distance inspired measure. Mentions are disambiguated by taking 40% of the most related candidate pairs and the most common pairs from this group are considered entities.

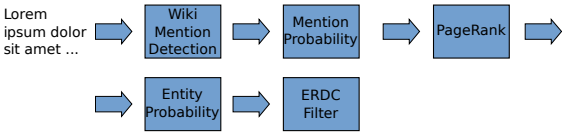

Figure 1: System configuration using several Transformers.

Document or query level disambiguation

To resolve entities, features of all candidates are used. One of the earliest work of entity linking [3] was based on similarity between document and candidate vectors (Wikipedia articles). For each candidate tf-idf vector was generated from its context and its categories. Candidate's score was computed as the similarity between document context vector and candidate vectors.

Corpus level disambiguation

Using features extracted from a corpus after entity linking is applied was proposed in [7]. Context around the same extracted entities can be grouped to global contexts to improve disambiguation in the next iteration. Expected entity count is used as a feature to detect systematic errors. There are several other approaches. One interesting example is in [9], where authors combine word sense disambiguation with entity disambiguation while utilising semantic networks.

3. OUR SYSTEM

Our system for entity recognition and disambiguation is based on DBpedia. Our knowledge base is based on Wikipedia and we use DBpedia uris for identification of entities.

We use a modular architecture which permits connecting various methods sequentially. It allows for quick trials using different configurations and combinations. Our basic module called "Transformer" receives a sentence object, which contains the text to be annotated, mentions and candidate entities. Transformer process the object and returns it back. For example, Transformer for mention detection finds all mentions in the text and add them to the object. A disambiguator assigns the candidate entities a score. There is an example how several Transformers are connected in Figure 1. This configuration is the one we used for final evaluation. Its description of the components is in the following section.

In the following subsections, we describe the two main components of the system - the mention detection and the disambiguation - and the final selection of entities to match the entity snapshot.

3.1 Mention detection and candidate selection

The mention detection is based on data provided by Wikipedia. We started by collecting labels for each entity - entity name and its redirects. By removing terms in brackets, such as in
S60_(software_platform), and tokenizing the remaining text we get a set of alternative names for the entity. By reversing the mapping $entity \rightarrow names$ we get a set of entities for the text.

The mention detection is restricted to such sequences of tokens that contain at least one capitalized letter, such as "eBay" or "Nokia E72". This restriction decreased recall, but highly increased precision. The mentions that consist only of stopwords are removed as well.

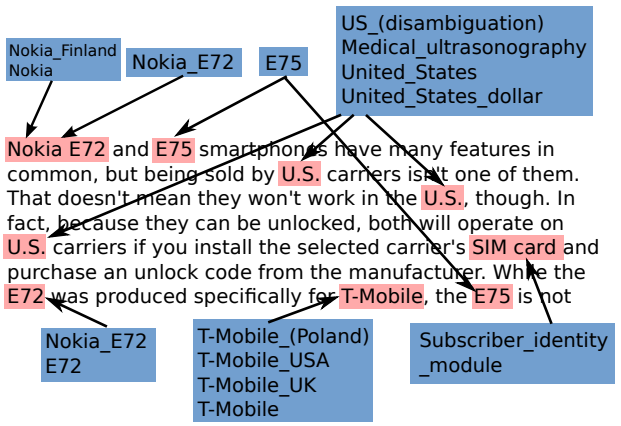

Figure 2: Sentence with mentions and entity candidates.

The text is processed sequentially, finding all the possible text mentions about entities. The length of matched text is limited to twenty words. For each mention we have a set of candidate entities that may be its target.

Tracing subsequent mentions of an entity

For each candidate entity we create a set of alternative names to be recognized as this particular entity. First, we add all words from the name of the entity (except the terms in brackets at the end of the name) and their capitalized version. Then we add all possible abbreviations made from the label. E.g. for "International Olympic Committee" we create a set of labels "International", "Olympic", "Committee", "IOC", "I.O.C" and "I.O.C." If any of these labels are among already found mentions in the text after the first occurrence of the entity, we add the International Olympic Committee as a candidate entity to that mention. For example, in Figure 2, the mention E72 in the last line has two candidate entities - E72 and Nokia_E72. Nokia_E72 has been added after the processing of the mention "Nokia E72" in the first sentence. Note that if there was a word "Nokia" in the remaining text, Nokia_E72 would be listed as a candidate entity as well. It may sounds illogical, but in case of mention "Richard Desmond" and candidate uri Richard_Desmond it is reasonable to mark consequent word Desmond as Richard_Desmond. The score of such added entity is boosted even in the case it is already in the list.

We limited this approach to one word from the label of the entity, but it may be reasonable to use more than one. In the training texts, there was following part (with emphasized text):

First Citizens BancShares Inc. of Raleigh, N.C., said it plans to ... *First Citizens* also owns ...

The "First Citizens" in the second sentence clearly points to the same entity as the full name in the first sentence. To mark all subsets of words from the entity label required too much processing time, therefore we searched for only one word from the label.

After we find a mention and it is followed by a word in brackets, e.g. "Immigration and Refugee Board (IRB)", we use the word as abbreviation of the longest mention before brackets.

Here follows options we have tested, but proved unsuccessful:

- Wikipedia disambiguation pages provide alternative labels as well, but they bring too much noise in the candidate set, making the job too difficult for the disambiguation.
- Using anchor text from wikipedia links. Its result is similar to the previous case.
- Adding label words and abbreviations as mentions. When we used the words from the entity label to create new mentions and not only to add the entity to existing mentions, it again brought too much noise even if we tried to filter the words by their idf.
- Not applying the restriction of one capitalized letter to already found mentions. E.g. when we found "Nokia", allow finding "nokia" as well.

3.2 Disambiguation

The disambiguation assigns a score to each candidate entity. We used a threshold to discard the entities with too low a score. The disambiguation uses three kinds of information - the co-occurrence of pairs of entities, "EntityProbability" the probability that the mention corresponds to the entity ($P(e|a)$, where e is an entity and a is an anchor) and "MentionProbability" the probability that the mention detection correctly assigns the mention to the entity ($P(e|M)$, M is mention detection from previous section). In the following sections, all three components are described in detail.

MentionProbability

MentionProbability is computed for each entity to estimate the probability that the mention found by mention detection really leads to the entity. The MentionProbability is computed as follows:

$$MentionProbability(e) = \frac{\#\text{ mentions found which lead to e}}{\#\text{ mentions found with candidate entity e}}$$

MentionProbability penalizes the entities with otherwise commonly used names, e.g. entity 19, which corresponds to year 19 A.D., has $MentionProbability(19) = 0.0002$. On the other hand $MentionProbability(\text{GSM}) = 0.41$.

MentionProbability can be viewed as apriori probability of mention detection detecting the right entity.

EntityProbability EntityProbability is a priori probability that a mention leads to an entity. It is computed from Wikipedia links and their anchors.

$$EntityProbability(e|a) = \frac{\#\text{ anchors a which lead to entity e}}{\#\text{ anchors a}}$$

The anchor a has to be the text of an anchor. For example $EntityProbability(.eu, \text{"eu"}) = 0.01$, which means that if "eu" is an anchor, it leads to entity .eu with probability 0.01. The probability of the anchor is distributed among several entities : $EntityProbability(\text{GSM}, \text{"gsm network"}) = 0.571$, $EntityProbability(\text{GSM\_services}, \text{"gsm network"}) = 0.142$ and $EntityProbability(\text{Network\_switching\_subsystem}, \text{"gsm network"}) = 0.285$. To complete the eu example, $EntityProbability(\text{European\_Union}, \text{"eu"}) = 0.95$.

Using entities co-occurrences

The disambiguation process uses a graph made of candidate entities, where two entities are linked, if they are often

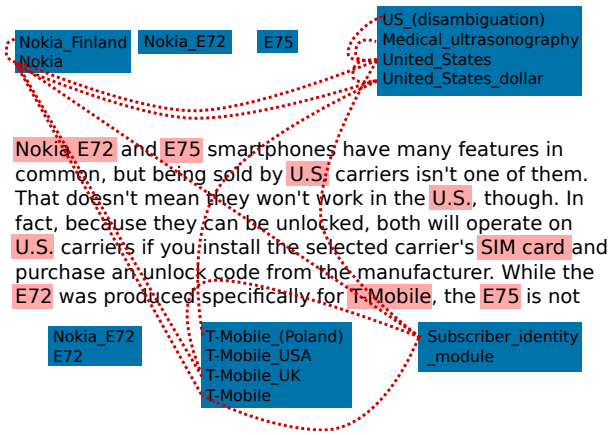

Figure 3: Sentence linked entity candidates.

used in the same context. The links between entities were extracted from Wikipedia articles - two entities are linked in one of the following cases:

1. The page of entity 1 has a link to entity 2.
2. There is a paragraph with a link to entity 1 and to entity 2.
3. The URI of the entities is the same.

These links are used to connect candidate entities in the sentence. We connect two entities, if their mentions are close enough and if there is a link found in Wikipedia. In Figure 3 is the former example with linked candidate entities. The links to the same entity are omitted. Note that Nokia_E72 or T-Mobile_(USA) have no links other entities than self-links. The figure also illustrates possible imperfections in links structure - there is a links between T-Mobile_(Poland) and United_States_dollar, which we did not expect and on the other hand Nokia_E72 is not linked with Nokia.

An example of such a graph is in Figure 4. The entities in the graph can be duplicated, as they can occur in more mentions. The clusters of the same entities are clearly visible as well the connections between similar entities. We point out that the Nokia_E72 is not linked to Nokia, because there is no link on Nokia_E72 page to Nokia and they do not occur in a paragraph together. On the other hand, it is linked to S60_(software_platform) and Symbian, which are linked to Nokia. The size of the nodes corresponds to the final score the entity got. Different sizes for the same entity can happen as the entity may be in different mentions, therefore in a different context.

For disambiguation, we used PageRank [2] on the graph of entities. The inspiration for using Pagerank for disambiguation comes from [1]. We tried several alternatives on how big the graph can be:

1. The nodes of the graph were the entities found as candidate entities to some mention.
2. Same as 1 and we add entities that are linked to at least k nodes in the graph.
3. We use the graph of all entities, but only the found entities emit the pagerank.

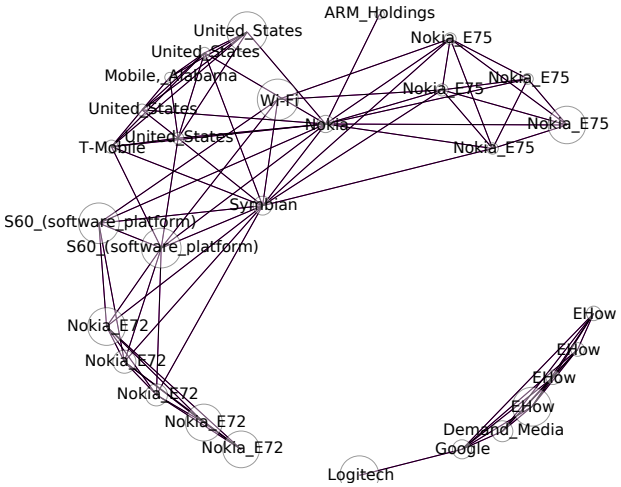

Figure 4: A sample graph of linked entities.

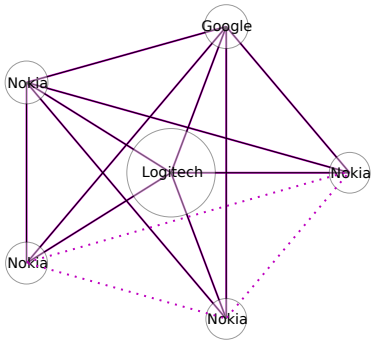

Figure 5: Graph of neighbours of Logitech with missing links dotted.

Finally, we used the first and the smallest version. The later ones were much slower and provided no increase in precision.

The PageRank algorithm was modified in following way. We wanted that entities that were more probable candidates emit more rank than the less probable entities, so instead of emitting a constant, entities emit their MentionProbability.

The weight of the link between the entities is omitted and we use constant damping factor. Only exception is the link between two entities with same uri, where the damping factor is divided by two. This weakens the "self-promotion" of an entity.

Next heuristic is to use "curvature", proposed in [4]. The curvature is defined as follows:

$$curvature(e) = \frac{\#\text{ triangles e participates in}}{\#\text{ triangles e could participate in}}$$

In Figure 5 is a graph of neighbours of Logitech. The number of triangles is 7 and number of possible triangles is 10, so $curvature(Logitech) = 0.7$.

After the PageRank algorithm, the entities has a score assigned. This score is then weighted by EntityProbability (a probability that the anchor of the mention leads to the given entity). This corresponds to "commonness" defined in [8].

There were again several dead-ends we followed:

- We tried to use some degree of the linkage between two entities, such as the relatedness [8], but the results were worse. The reason behind this is that the number of entities co-occurrences is small and the weight of the link is not significant.

- The first approach we used was to create a textual context of an entity and match that context to the text around a mention. From tf-idf, through okapi, mutual information, entropy, information gain and to compression distance, none of these approaches contributed to performance of the system and they required heavy computational cost.

- Another approach is to use Viterbi algorithm [11], used for named entity disambiguation in [5]. This approach performed worse than the PageRank, probably because the training data was not big enough. Viterbi algorithm finds the most probable entities in the sequence of mentions, but with a lot of mention overlaps it was not performing well.

- Apart from EntityProbability, we computed the inverse probability $P(a|e)$ as well. It did not result into performance increase.

3.3 Mapping to Freebase entities

The entities found have to be mapped to Freebase and the results are limited to the entity snapshot provided by the organizers. During steps before, we used all entities from dbpedia and we included overlapping entities. Now we need to filter out those that are not in the snapshot and leave only the longest mention. Though it first seemed straightforward, it turned out to be a complex problem.

We often find an entity which is not in the snapshot that overlaps a mention of an entity in the snapshot. E.g. for a query "obama family tree", we find "obama", "obama family", "family tree" and "tree" as entity mentions. But entity Family_of_Barack_Obama, which is mapped to the mention "obama family", is not in the snapshot. If we discard the whole mention, we would loose the mention about Barack_Obama as well.

We also encountered cases such as this: "cherry tomato seed kit". We mapped tomato to entity Tomato. But Tomato is not in the snapshot, so we take the next best entity from the snapshot, which happens to be Tomato_(musician). Regarding the score, Tomato got 2.91 and Tomato_(musician) only 0.002. Therefore if the best entity from snapshot has much lower score than the best winning entity, we discard this mention.

Finally, the last border case is the following : "Symbian 9.3 series 60". Among others there are two overlapping mentions "3 series" as BMW_3_Series and "Series 60" as S60_(software_platform), but one is not a submention of the other. In this case, we take the mention longer by word count. If the number of words of mentions is equal, then we take the mention, whose winning entity has higher score. In this case, the S60_(software_platform) has a score of 2.96 and BMW_3_Series has 0.64, so the S60_(software_platform) wins.

Nokia E72 and E75 smartphones have many features in common, but being sold by U.S. carriers isn't one of them. That doesn't mean they won't work in the U.S., though. In fact, because they can be unlocked, both will operate on U.S. carriers if you install the selected carrier's SIM card and purchase an unlock code from the manufacturer. While the E72 was produced specifically for T-Mobile, the E75 is not

Figure 6: Two overlapping windows of size 6.

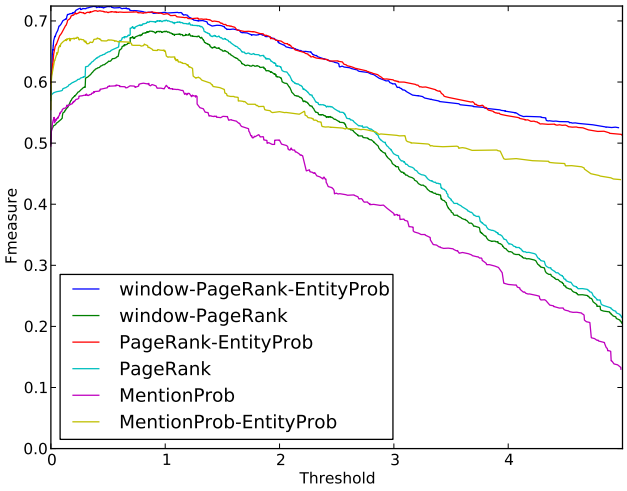

Figure 7: F-measure based on threshold for different configurations.

4. PERFORMANCE ON ERDC DATA

4.1 Specifics of the Long Track system

The Long Track texts were taken from web pages, the length of the documents ranged from 100 words to over 1500. For large texts, creating the whole graph is intractable, as all pairs of entities has to be processed and the construction of such a big graph is computationally expensive. Therefore a sliding window on mentions was used - the graph is created from candidate entities of 20 mentions at a time. The overlap was 10 mentions and the score was averaged from the two windows which used the same mention. The example of the window is in Figure 6.

4.2 Our performance on the Long Track

The organizers provided us with a set of 51 documents annotated with entities. These annotations were not final, but it was a guidance to quickly estimate the performance of the system. The performance estimate from our evaluation were different from a case when we run the proper evaluation on the ERDC website, but usually if the system improved offline, it improved online as well. The online test set used 101 documents. We will use our offline metric in the following text and compare it to the online testing at the end of the section.

We started with a simple mention detection based on Wikipedia and the focused on improving the disambiguation. After we were happy with disambiguation performance, we got back to mention detection and tried to improve its recall by introducing the tracing entities mentions by a word from

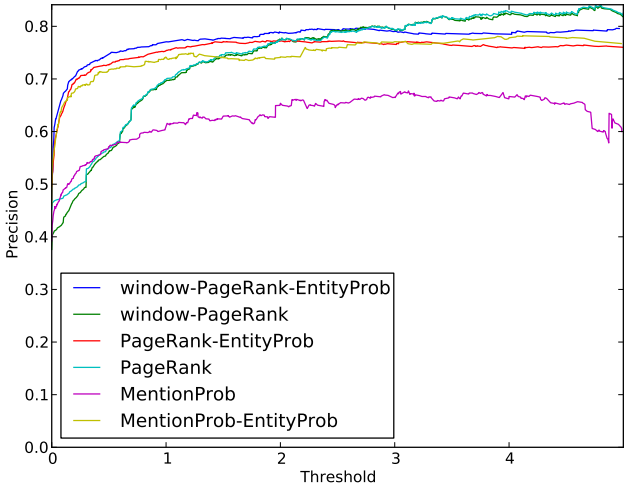

Figure 8: Precision based on threshold for different configurations.

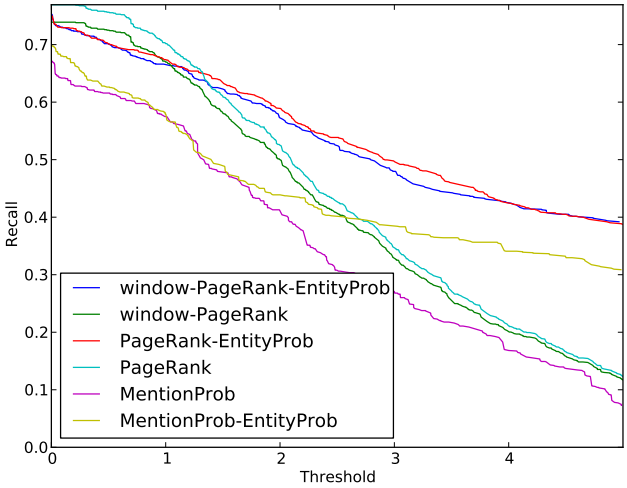

Figure 9: Recall based on threshold for different configurations.

title or its abbreviation. In the following text, we will discuss the disambiguator tuning when using the best mention detection we used in final system.

First, we used a simple heuristic - using MentionProbability as disambiguation score. We got f-measure 0.5850, precision 0.5778 and recall 0.5924.

Then we weighted the score by EntityProbability and the scores improved to f-measure 0.6605, precision 0.6708 and recall 0.6505. Up to here, the time to process the 50 texts was 11ms.

The PageRank on the graph of entities was introduced, achieving f-measure 0.7235, precision 0.7538 and recall 0.6956. Using PageRank required some parameter tuning. E.g. the number of iterations is 10. If we increase it to 20, the performance is f-measure 0.7182, precision 0.7415 and recall 0.6964.

Unfortunately, the construction of the graph was too demanding and the time for processing the dataset increased to 150ms. Hence we used the sliding window described in Section 4.1. After that, the time dropped to 72ms and we did not get any timeouts from the online testing. Using window, the performance changed to f-measure 0.7272, precision 0.7671 and recall 0.6912.

Final heuristic was to use the curvature, which did not increase the processing time significantly and improved the performance to final f-measure 0.7305, precision 0.7659 and recall 0.6982.

This system configuration had f-measure 0.7509, precision 0.7796 and recall 0.7242 on the train set using online evaluation. Both the precision and the recall were considerably higher - this may be due to the fact that the rules stated that the correct annotation is when the right mention and the mention provided by the system "overlap", without further specification, but our offline evaluation required precise match.

The performance on the final testing set of was f-measure 0.7193, precision 0.7928 and recall 0.6583. Despite being much worse than the training set, it took us to third position by a hair's breadth (the fourth team had f-measure 0.7137).

There are three graphs showing f-measure, precision and recall on the vertical axis and threshold on the x axis in Figures 7, 8 and 9. Each line corresponds to a variant of our system. The threshold is used to discard all entities with lower score than the threshold. High threshold results in low recall and high precision and vice versa.

4.3 Specifics of the Short Track system

Web search queries are characteristic by its specific language and type of writing. The queries are often unformatted, it is common that they do not contain punctuation and/or diacritics and one could not expect even the right usage of capital letters. This last fact is quite important for our task, because most named entities begin with a capital letter, what could simplify the mention searching.

Misspells are common as well. Resulting problems could be solved by using query correction system but for our purpose we were satisfied with expected common errors contained in mention detection data from Wikipedia.

The most important difference from the ERD for texts lies in the possible ambiguity of detected queries. It is due to smaller extent of the given context. In these cases it was necessary to provide all possible correct disambiguations. The treshold for Short Track had to be set according this need. The other required modification was additional punishing of low score disambiguations in cases when another candidate with much higher score was present. This feature helped in cases when the context provided enough support for one of possible candidates correctly striking-out the others.

We made experiments with numbers of different set ups similarly to Long Track task. However we did not encounter any problems with time complexity as the texts analysed were much shorter. The main problem during the learning process was the amount of testing examples. The organizator provided us with set of 90 annotated queries, and similar amount (100) was used to determine quality of contending systems. This was a bit harsh while possible improvements of the system was hardly observable (e.g the number of all annotated entities for 90 queries was only 61). Similarly an improvement that was able to fix one annotation gave us almost 1% boost on f-measure. However similar improvement

was not expected on bigger amount of data. Luckily the testing set used was later extended to 500 queries, which gave us another possibility to systematicaly improve the settings of our system without worries about possible overfitting. The final setup we used was the same as for the Long Track, except for using unmodified PageRank on graph (without sliding window).

We were able to reach 0.6260 f-measure on regular test set and in the finals the number was improved to 0.6693.

5. CONCLUSIONS

ERD challenge gave us a great opportunity to enhance our work on entity recognition and disambiguation we were currently developing in Seznam.cz. Challenge of this kind always pushes the state-of-the-art performance further, let us mention the Netflix prize which accelerated the research in collaborative filtering movement.

Although this paper did not introduce any ground-breaking theoretical results, we believe that the actual performance of the system motivated by simple, logical heuristics, may inspire researchers in ERD. This contrasts with Netflix prize, which was won by a heavy, black-box, machine learning and ensembles of classifiers. Actually, we were surprised that the system without heavy parameter tuning and machine learning performed this well.

The system described has a modular architecture permitting easily using different configurations and methods in sequence. There is a few dozens from which we chose the best combination for the challenge. This configuration was thoroughly described in the paper, also with the information how each of the component increased the performance of the system. We omitted detailed description of other components we did not use in the challenge.

The lack of machine learning represents also a room for improvement. Using of large data sets like Clueweb [6] with deep learning and statistical processing, creates a potential to bring the ERD to another level. For example, having more entities co-occurences, we may be able to use the weight of the link in the PageRank.

Also using more than one disambiguator and learning a classifier based on the scores from disambiguators output is an interesting way to go. We did many attempts to make use of the textual contexts, but with no results. Using this information as a feature to a full classifier should provide new information. Our preliminary trial were not successful, but we still think there is a potential.

6. ACKNOWLEDGMENTS

We would like to thank the organizers of ERD challenge for a great opportunity and great organization.

7. REFERENCES

[1] E. Agirre, O. L. de Lacalle, and A. Soroa. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84, 2014.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998.

[3] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716, 2007.

[4] J.-P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. pages 5825–5829, 2002.

[5] R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. Named entity recognition through classifier combination. In *In Proceedings of CoNLL-2003*, pages 168–171, 2003.

[6] E. Gabrilovich, M. Ringgaard, and A. Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), June 2013.

[7] T. Lin, Mausam, and O. Etzioni. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 84–88, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[8] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *In Proceedings of AAAI 2008*, 2008.

[9] A. Moro, A. Raganato, and R. Navigli. Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244, 2014.

[10] U. C. D. Program. UCDP: Definitions, 2014. [Online; accessed 27-June-2014].

[11] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, 1967.

[12] C. von Clausewitz. *On War*. Project Gutenberg, 2006.

[13] Wikipedia. Sport, 2014. [Online; accessed 27-June-2014].