Extending a multilingual Lexical Resource by bootstrapping Named Entity Classification using Wikipedia's Category System

Johannes Knopp
KR & KM Research Group, Department of Computer Science
Universität Mannheim, B6 26, 68159 Mannheim, Germany
johannes@informatik.uni-mannheim.de

Abstract

Named Entity Recognition and Classification (NERC) is a well-studied NLP task which is typically approached using machine learning algorithms that rely on training data whose creation usually is expensive. The high costs result in the lack of NERC training data for many languages. An approach to create a multilingual NE corpus was presented in Wentland et al. (2008). The resulting resource called *HeiNER* describes a valuable number of NEs but does not include their types. We present a bootstrap approach based on Wikipedia's category system to classify the NEs contained in HeiNER that is able to classify more than two million named entities to improve the resource's quality.

1 Introduction

For tasks in information extraction NERC is very important and often supervised machine learning approaches are used to solve it, e.g. Bender et al. (2003) or Szarvas et al. (2006). In *A survey of named entity recognition and classification* David Nadeau and Satoshi Sekine conclude:

> "When supervised learning is used, a prerequisite is the availability of a large collection of annotated data. Such collections are available from the evaluation forums but remain rather rare and limited in domain and language coverage" (Nadeau and Sekine, 2007)

To overcome the problem of limited language coverage, Wentland et al. (2008) started to create the multilingual *Heidelberg Named Entity Resource (HeiNER)*. In more than 250 languages, HeiNER lists Wikipedia (WP) articles that describe a named entity (NE), in 16 of those languages it contains a collection of textual contexts a NE was unambigiously mentioned in. Those contexts provide useful training material for NE classification, thus the goal of this work is to add NE types to HeiNER's entries.

Unlike the widely used machine learning approaches to NERC our classification method relies only on WP's category system and thus does not need any language specific information. The idea is to first determine sets of WP categories to identify each NE type. After that, these sets are used to initialize a bootstrapping algorithm that identifies the types for unclassified NEs. NE types follow the CoNLL definition presented by Sang (2002): person (PER), location (LOC), organization (ORG) and miscellaneous (MISC).[1] The CoNLL types were chosen because HeiNER's evaluation was based on the CoNLL types.

The following sections reveal details about HeiNER (section 2), describe the bootstrap approach of NE classification with WP categories (section 3) and show the results in the evaluation section (section 4).

2 HeiNER

As this work builds upon the Heidelberg Named Entity Resource (HeiNER), we will describe the data that HeiNER provides and how they were created to give the reader an idea about their quality and structure.

HeiNER is a multilingual collection of *named entities* along with *disambiguated context excerpts* and a *disambiguation dictionary* that maps proper names to a set of NEs the proper names may refer to. The resource was created automatically from Wikipedia relying on (i) the heuristic presented in Bunescu and Paşca (2006) to recognize English Wikipedia articles that denote a NE and (ii) Wikipedia's link structure.

[1] cf. http://www.cnts.ua.ac.be/conll2003/ner/annotation.txt

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 35–43,
Chiang Mai, Thailand, November 8-12, 2011.

```
<transDict>
<namedEntity id='2134'>
<an>Organizazión d'as Nazions Unitas</an>
<bs>Ujedinjeni narodi</bs>
<ga>Nisiin Aontaithe</ga>
<gl>ONU</gl>
<hu>Egyesült Nemzetek Szervezete</hu>
<lb>Vereent Natiounen</lb>
<nds>Vereente Natschonen</nds>
<tr>Birleşmiş Milletler</tr>
<en>United Nations</en>
...
</namedEntity>
</transDict>
```

Figure 1: Example of the entry for "United Nations" in the translation dictionary

First, the NER heuristic based on uppercase letters generated a list of English WP articles that denote a NE. This method created more than 1.5 million NEs with a precision of 95% [2]. With help of WP's interlanguage links the available translations for every NE were added to the list resulting in the *translation dictionary* shown in figure 1. All of the more than 250 languages available in WP were considered to create the NE translations.

As the NE articles in WP are known from the first step, the disambiguation dictionary is built afterwards using disambiguation and redirect links to map proper names to NEs. Finally the context dataset is created for every NE by storing the paragraphs they are unambiguously mentioned in. This was done for 16 languages. An excerpt of the context dataset is shown in Figure 2 below.

```
<dataset  neID='2134' lang='en'
          neStr='United Nations'>
<context id='0'>
 <surfaceForm>United Nations</surfaceForm>
 <leftContext>
 The World Health Organization (WHO) is a
 specialized agency of the
 </leftContext>
 <rightContext>
 (UN) that acts as a coordinating
 authority on international public health.
 </rightContext>
</context>
</dataset>
```

Figure 2: Excerpt from the English context dataset for the NE "United Nations"

The NEs together with disambiguated contexts in different languages can be considered useful data for NE disambiguation, classification or machine translation (e.g. Federmann and Hunsicker (2011)).[3] For this paper the heuristics to create the list of English NEs were run on the more recent WP dump of November 3rd 2009 and resulted in a total of 2,225,193 found NEs compared to 1,547,586 NEs reported in the original paper. The difference is solely caused by the natural growth of Wikipedia.

3 A Bootstrap Approach to NE Classification with WP Categories

As described in Section 2 HeiNER presents a lot of context information of NEs. To release the full potential of the multilingual data the NEs need to be annotated with their respective type.

Instead of using a classical NER system this work concentrates on a language agnostic approach that is based on WP's category structure which is not only suited for NER but can be used for other classifications based on WP categories as well. In short, the idea is to identify WP categories that correspond to a NE type and then use those categories to classify NEs that are placed in those *typed categories*. The categories can be interpreted as a signature or footprint of a NE type. The method outline is as follows: First, for every NE type a list of seed categories is created manually. It is enhanced by taking two levels of subcategories into account. The resulting lists of type specific categories are used to classify the articles in HeiNER by looking up if they are placed in one of the seed categories and assigning the respective type. The steps are illustrated in figure 3.

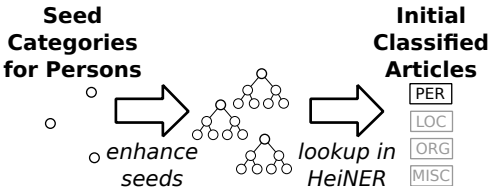

Figure 3: The manually chosen and enhanced seed categories generate the initial list of classified articles. The illustration shows the method for PER, it works in the same way for the other categories.

This leaves most of the NEs in HeiNER unclassified, but the initially classified NEs can be used for the bootstrapping solution that is visualized in figure 4: For every NE type, a NE type vector

[2] Read Wentland et al. (2008) for more details.

[3] HeiNER is available for the scientific community at http://heiner.cl.uni-heidelberg.de/

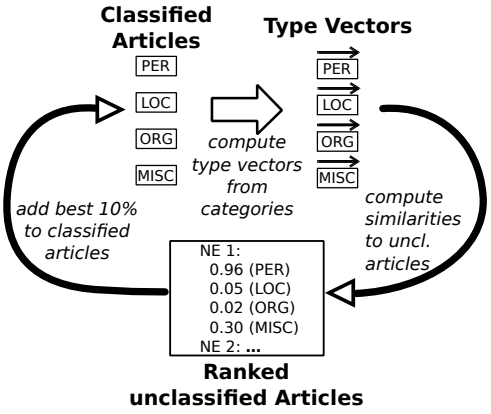

Figure 4: Bootstrapping loop to classify articles.

based on categories is built by looking up all categories of the now classified articles and counting them for each type. The articles are then classified by computing the similarity between their category vector and the four NE type vectors and choosing the most similar one. This is done in ten iterations where each step updates the type vectors with the new classified articles. The only manual work needed is collecting the seed categories. This can be applied in any language that is available in WP. We use the English version because it is by far the largest edition. Also note that the seeds define the result of the classification. More fine grained types like *politician* or *entertainer* (cf. (Fleischman and Hovy, 2002)) could be easily implemented by choosing other seeds.

After this broad overview the subsections present a more detailed description of the approach. For that we introduce the notation scheme used in this paper:
The set of NE types $t \in T$ consists of persons PER, locations LOC, organizations ORG and miscellaneous $MISC$.
C denotes the set of all categories in the English Wikipedia. Single Categories that are mentioned in the text are written in SMALL CAPS.

3.1 Generating Seed Categories

For every NE type the seed categories hold a set of WP categories such that any NE article that is placed in one of them is considered to be of the type the category is associated with. Because the classification method relies on the seeds' quality they have to be annotated manually. The goal is to find categories that are broad enough to classify as many NEs as possible but also are accurate in order to avoid incorrect classifications.

To find the best seed categories for the NE types person, location, organization and miscellaneous, we started to randomly pick NE articles belonging to one type, then inspect the categories it is placed in and move up in the category tree by following supercategories until the topic range of a category gets too broad for unambiguous classification. The broad-but-accurate categories are added to the seed set of the respective type. Because the subcategories can be considered to be useful for the classification process, we add two levels of subcategories to the initial seed list. The restriction to two levels of subcategories is needed to avoid adding noise, because WP's category system is a graph, not a tree.

An example for the manual creation of seed categories might help at this point: if we are interested in the NE type person, we start with a random WP article about a person, e.g. Jimmy Hendrix. We always follow the most promising supercategories which leads to the following chain:
1960S SINGERS $\Rightarrow$ SINGERS BY TIME PERIOD $\Rightarrow$ PEOPLE BY OCCUPATION AND PERIOD $\Rightarrow$ PEOPLE BY OCCUPATION $\Rightarrow$ PEOPLE

The accuracy of each category is checked by inspecting subcategories and articles belonging to it. The category PEOPLE has a subcategory BIBLIOGRAPHY which deals with biographical books. Thus, PEOPLE itself is not accurate enough to find persons. Still most of the subcategories of PEOPLE like PEOPLE BY OCCUPATION or PEOPLE BY RELIGION are added to the seed categories of NE type person.

As a result there are 15 seed categories found for the type person. The same was carried out for the other NE types. All seed categories together with two levels of subcategories form the set of typed categories C_t. The results can be seen in table 1.

The number of seed categories does not necessarily correlate with the number of found subcategories: The types PER and LOC have the same count of seed categories, but C_{PER} is almost 3.5 times bigger than C_{LOC} and has about 1,500 categories more than C_{ORG} which started with 75 seed categories. An explanation would be that persons are supported well and have a very fine grained categorization while locations can be described with a smaller set of categories. C_{MISC} remains in between the others with 4,747 subcategories.

type t	seed categories	sub-categories	typed categories C_t
PER	15	9,625	9,640
LOC	15	2,783	2,798
ORG	75	8,033	8,108
MISC	27	4,747	4,774

Table 1: Numbers of categories found for each NE type derived from seed categories.

3.2 Initial Named Entity Classification

Starting from the enhanced seed categories the initial list of classified NEs can be created easily. Just iterate over every article in HeiNER and check if it is placed in C_t. If this is the case the article can be considered to be of type t and hence is added to the set of classified NE articles NE_t. If more than one type was found for an article it is left unclassified. The results of this initial classification are shown in table 2.

To point out the generative power of the categories the last row shows the "productivity ratio" $\frac{NE_t}{C_t}$ of each category. The earlier assumption that there are more articles of type PER than others is supported by the fact that more than half million NEs could be initially classified and also by the number of articles found per category. This cannot be solely based on the superior count of PER categories because the number of ORG related categories is not that far behind, though NE_{ORG} is about 4 times smaller than NE_{PER}. Also the PER related categories are about five times more productive than the ones related to MISC. In other words, most of WP's contributors write articles about NEs of the type PER and categorize them studiously. The quality of the results will be discussed in the evaluation in section 4.

Type t	C_t	NE_t	$\frac{NE_t}{C_t}$
PER	9,640	502,173	52
LOC	2,798	41,539	15
ORG	8,108	128,433	16
MISC	4,774	47,887	10

Table 2: Number of classified articles derived from seed categories. The last row shows the rounded average classification produced by each category.

3.3 Type Vectors & Bootstrapping

After the initial classification step we can remove the 720,032 classified articles from the NE list with 2,224,472 entries leaving 1,504,440 yet to classify articles. As the presented method relies on categories 7,033 articles without any categorization are removed too which results in a final list of 1,497,407 NEs that need to be classified in the bootstrapping process.

As explained earlier the categories of the classified articles are used to build a NE type vector consisting of categories associated with NEs of a certain type. The categories of classified articles form the dimensions of the type vectors, their counts define the length in that dimension. The algorithm in figure 5 shows how the vector is created. Note that for the NE type vector all categories are taken into account and not just the ones pointing to NEs that were used in the initial classification step. The intuition behind this is that the aggregated categories form the footprint of a type even if not each of them points to a NE.

```
def compute_vector(NE_t):
  #store vector as a dictionary
  category_vector = {}
  for article in NE_t:
    for c in article.catgories:
      if category_vector.has_key(c):
        category_vector[c] += 1
      else:
        category_vector[c] = 1
  return category_vector
```

Figure 5: Python-Pseudocode algorithm of a function to build the category vector. The vector is stored in a dictionary where the category name is the key and the count its value.

The algorithm is applied to each NE type in NE_t, the results are shown in table 3. The dimensions of the vectors in the third row show the number of unique categories. The fourth row represents the overall count of categories in the articles and the last row shows the average number of categories per article. Again we can see that PER is categorized in more detail while LOC and ORG have a similar ratio. MISC has the lowest categorization rate. We expect our method to work best with articles that are placed in many categories.

The type of an unclassified NE article is determined by converting its categories into a vector, computing similarities to the type vectors, and as-

type t	NE_t	dimensions	category count	categories per NE_t
PER	502,173	132,098	4,037,634	7.86
LOC	41,539	35,880	228,468	5.08
ORG	128,433	72,184	694,523	4.94
MISC	47,887	33,110	229,438	4.33

Table 3: Statistics for the NE type vectors that are created for NE_t.

signing the type with the highest similarity score. As categories can either be present or not the category vector of an article is binary. In order to verify the general approach we classify the NEs in two setups using different similarity measures, *cosine similarity* and *Dice's coefficient*:

$$cosine(\vec{x}, \vec{y}) = \frac{\sum_{k=1}^{n} x_k y_k}{\sqrt{\sum_{k=1}^{n} x_k^2} \cdot \sqrt{\sum_{k=1}^{n} y_k^2}}$$

$$dice(\vec{x}, \vec{y}) = \frac{2 \cdot \sum_{k=1}^{n} (weight_{xk} \cdot weight_{yk})}{\sum_{k=1}^{n} weight_{xk} + \sum_{k=1}^{n} weight_{yk}}$$

Cosine similarity computes the angle between the two vectors taking only the directions of type vectors into account and not their length. Because there are no negative categorizations the resulting similarities range between zero and one. The Dice's coefficient includes the count of shared elements in relation to all elements that are not zero. It considers the weights of the vectors by multiplying the shared elements[4]. The factor 2 keeps the result range between zero and one.

In the bootstrapping phase HeiNER's unclassified NEs are classified as just described. In 10 iterations the 10% with the highest similarity values are added to their respective set NE_t and the type vectors are updated before the next 10% are classified. Figure 6 shows the process for cosine similarity and figure 7 for Dice's coefficient.

For each NE type the tables list the exact counts of how many NEs were added in each of the 10 iterations. The bar plots beneath the tables visualize these data by stacking the counts of each type in every iteration. As the sum is always 10% of the initially unclassified data the bars have the same length. The exception at iteration 10 stems from the fact that articles that do not share a category with any of the type vectors cannot be classified. The difference between the last Dice and cosine bar is a result of the different classification decisions made in the bootstrapping process.

[4]As we multiply with a binary vector we just decide whether to add the value of the non-binary vector at that position or not.

run	**PER**	**LOC**	**ORG**	**MISC**
initial	**502,173**	41,539	128,433	47,887
		Cosine		
1	3,999	**120,641**	23,469	1,631
2	1,216	11,456	42,997	**94,071**
3	1,414	**56,725**	38,220	53,381
4	33,664	11,763	39,064	**65,249**
5	50,990	10,690	17,511	**70,549**
6	44,166	24,131	22,569	**58,874**
7	14,924	39,565	33,347	**61,904**
8	4,482	45,417	37,201	**62,640**
9	3,392	38,138	38,711	**69,499**
10	4,057	26,395	38,719	**60,913**
Bootstrap	162,304	384,921	331,808	**598,711**
Total	**664,477**	426,460	460,241	646,598
Plus	32%	927%	258%	**1250%**

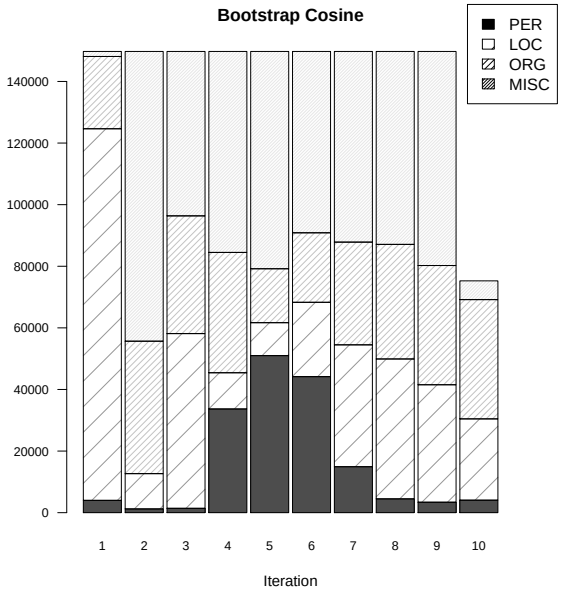

Figure 6: Bootstrapping using cosine similarity. The bar plot shows the visualization of the NE type classifications in the table above.

Inspecting the results we can see that the lion's share in the first iteration in both setups is classified as LOC. This indicates that many locations were missed by the enhanced seed categories, but the type vector allowed to find the missed NEs. Following iterations do not show a bias towards LOC which supports this analysis. Nevertheless cosine similarity seems to be biased towards MISC because on average about 60,000 articles are added to this type per iteration resulting in the biggest gain in 8 of the 10 iterations. This could be caused by cosine similarity's ignorance of weights in the type vector thus preferring articles that share

many categories with a type vector over articles with less but higher weighted categories. MISC might have thematically wide spread categories supporting that effect. However, the bias towards that type cannot solely be based on this property, because the initialized vector is the one with the least dimensions in comparison to the others.

Bootstrapping using the Dice's coefficient tends to be biased towards LOC and ORG, the former showing an overall gain of 1,308 percent[5]. In four of the iterations ORG wins the majority of new classified articles, LOC is in advantage in five of the iterations leaving PER one major gain in the fifth run. Because Dice's coefficient takes the counts of categories into account, it is likely that the unclassified articles are placed in some of the categories that have high values for LOC and ORG.

The count of articles added to PER develops remarkably similar for both measures. They start with few new articles in the first three iterations, rise to many more additions in steps four, five and six to slow down again in the left iterations. In both cases eventually PER is the NE type with the least added articles (cf. lines "Bootstrap"), but still the biggest count when summing it up with the initial count (cf. lines "Total"). No other named entity type shows such a strong correlation between the two different similarity measures. This indicates that most of the articles were already classified in the initialization proving the seed categories for that type to be of high quality.

In summary, both bootstrapping setups are able to classify almost all of the unclassified NEs, but differ a lot in their results with the exception of the type PER.

4 Evaluation

Before the bootstrapping phase an evaluation set of NEs was created and excluded from the process. It consists of NEs of each type: 295 PER, 192 LOC, 110 ORG and 122 MISC entries that were annotated manually by one annotator. Both setups are evaluated by classifying the NEs in the same way as in the bootstrapping and investigating the precision of the results.

[5]This growth is narrowed a little bit by the fact that it started with the smallest count of articles.

run	PER	LOC	ORG	MISC
initial	**502,173**	41,539	128,433	47,887
Dice's coefficient				
1	5,271	**137,051**	6,406	1,012
2	17	25	**138,578**	11,120
3	1,266	58,780	**65,593**	24,101
4	36,595	16,952	**56,017**	40,176
5	**67,975**	31,508	25,819	24,438
6	38,196	**56,745**	45,219	9,580
7	16,166	**67,458**	54,813	11,303
8	8,969	**67,890**	52,944	19,937
9	5,581	**65,655**	46,860	31,644
10	5,751	41,301	**56,864**	26,323
Bootstrap	185,787	543,365	**549,113**	199,634
Total	**687,960**	584,904	677,546	247,521
Plus	37%	**1,308%**	427%	417%

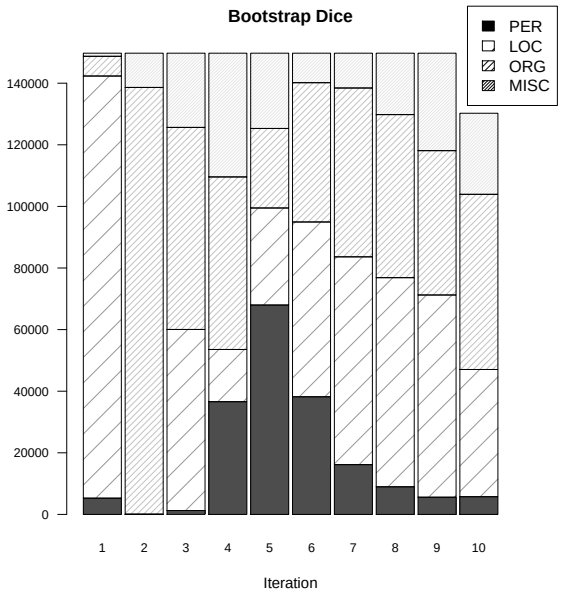

Figure 7: Bootstrapping using Dice's coefficient. The bar plot shows the visualization of the NE type classifications in the table above.

4.1 Initial type vectors

The confusion matrix in table 4 shows the results using the type vector from the initial NE classifications. The rate of correct classifications varies from 35.25% (MISC, Dice's coefficient) to 81.02% (PER, Dice's coefficient). It is not surprising that PER is the best performing named entity type when we remember the earlier statement that articles of that type are categorized with high detail and that this NE type has by far the highest count of instances after the initialization. This is underlined by the fact that almost no instances were classified incorrectly as a person in the other evaluation sets. Consequently, there is no much confusion between persons and other NE types.

Eval. set	PER	LOC	ORG	MISC	UNCL
Cosine					
PER (295)	78.64% (232)	5.76% (17)	8.47% (25)	6.44% (19)	0.68% (2)
LOC (192)	0.0% (0)	60.42% (116)	10.94% (21)	7.29% (14)	21.35% (41)
ORG (110)	0.91% (1)	15,45% (17)	**67.27% (74)**	8.18% (9)	8.18% (9)
MISC (122)	0.82% (1)	8.2% (10)	**38.52% (47)**	37.7% (46)	14.75% (18)
Dice's coefficient					
PER (295)	**81.02% (239)**	6.1% (18)	7.8% (23)	4.41% (13)	0.68% (2)
LOC (192)	0.0% (0)	**64.06% (123)**	9.9% (19)	4.69% (9)	21.35% (41)
ORG (110)	1.82% (2)	19.09% (21)	64.55% (71)	6.36% (7)	8.18% (9)
MISC (122)	3.28% (4)	9.84% (12)	36.89% (45)	35.25% (43)	14.75% (18)

Table 4: Confusion matrix for the CoNLL named entity types. Members of evaluation sets for every type were classified by computing similarities to the initialised named entity type vectors. The overall highest values (cosine and Dice similarity) are marked as boldface. The percentages show the fraction of the absolute numbers that are given in the first row, the numbers in braces show the absolute numbers.

Considering that 21.35% of the articles were left unclassified, only 18.23% (cosine) and 14.59% (Dice) of LOC were explicitly classified wrong. Unclassified articles occur if none of the instances in the evaluation set LOC has categories that can be found in any of the NE type vectors. This could either mean that the seed categories for this type were not chosen broad enough or that articles of type LOC are placed in categories that are wide spread over WP's category graph and cannot be grouped easily. The bootstrapping results indicated that the former case is more likely. ORG are classified correctly with a chance of 67.27% (cosine) and 64.55% (Dice) leaving an error rate of 24.55% (cosine) and 27.27% (Dice). Cosine outperforms the Dice's coefficient in this class.

The CoNLL definitions of MISC do not seem to correspond well with WP categories. For the evaluation set of type MISC more instances were classified as an organization in both setups. That indicates a high probability to confuse members of MISC with LOC which is not that surprising, recalling that the definition of this type is "words of which one part is a location, organization, miscellaneous or person"(Sang, 2002). Further investigation would be necessary to judge whether type overlaps are just caused by incorrect classifications or if the articles really do belong to that class and maybe should be allowed to be classified as both MISC and LOC. For example a book that has a location in its title like *The Restaurant at the End of the Universe* could benefit from a double classification because depending on the context it may serve as one or the other.

The results of the initialization step show that in general the MUC-6 named entity types(Grishman and Sundheim, 1996) PER, LOC and ORG can be classified with this approach reasonably well with 60.42% (LOC, cosine) as lower and 81.02% (PER, Dice) as an upper bound. This does not work out as well for MISC, but still the lower bound of 35.25% (Dice) beats a baseline with randomly assigned types that would result in 25% correct classifications. Thus, the initially constructed type vectors are useful for NEC of WP articles. At this time it is not possible to say which of the similarity measures returns better results.

4.2 Bootstrapping Iterations

To evaluate the iterative classification phase we used the resulting type vectors of every step to classify the evaluation set and again analyze the percentage of NEs that were classified correctly. [6]

Figure 8 shows the results per iteration for each type and setup. The continuous line represents cosine similarity while the dashed line represents Dice's coefficient. To see which setup works best compared to the other the different lines marked with the same symbols must be compared. The lines point out the development of the quality of the type vectors.

After every iteration the type vector is refined which should improve classifications. However, because every classification step only incorporates the best or most certain 10% of unclassified NEs leaving the less clear NEs unclassified, the preci-

[6]Because the annotated data represent only a fraction of the whole data we cannot provide reliable recall results.

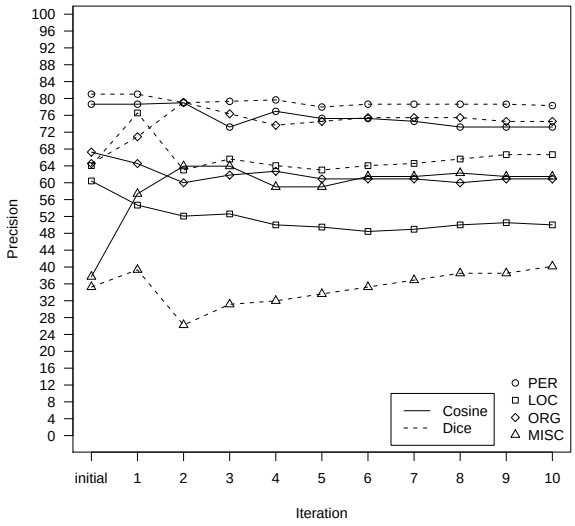

Figure 8: Precision of the classification for the iterations in the bootstrapping phase.

sion is expected to decrease in later iterations due to introduced noise. Thus a stable line indicates a successful approach.

If we ignore MISC for a moment, the cosine setup has an overall decrease in precision relative to their starting point while the Dice setup is fairly stable or even better. The difficulty of representing the MISC type with WP categories seems to be the reason for its different behaviour, the broad choice of categories creates the bias of the cosine method. Dice's coefficient is more robust and seems to avoid that noise making it more suitable for the task. This can be seen after the first iteration: As discussed in section 3.3 the biggest fraction was classified as LOC. While the precision of Dice's coefficient increases by more than 10% in this iteration the precision of the cosine setup drops more than 5% which implies that many NEs were classified wrong. Finally, the best results after bootstrapping are:

- *PER* – Dice 78.31% (cosine 73.22%)
- *LOC* – Dice 66.67% (cosine: 50%)
- *ORG* – Dice 74.55% (cosine: 60.91%)
- *MISC* – cosine 61.48% (Dice: 40.16%)

Dice coefficient performs better than cosine similarity for three out of four NE types, which implies that taking statistical evidence into account improves the performance of the classification. The numbers indicate that cosine similarity beats Dice coefficient at the classification of *Miscellaneous* because it is biased.

5 Conclusion

In this paper we have shown a language-agnostic method to classify more than two million NEs in the multilingual lexical resource HeiNER (Wentland et al., 2008) in two steps, adhering to the CoNLL definition of NEs (Sang, 2002; Sang and Meulder, 2003) relying on structural information only. First, we initialized 700,032 classified NEs utilizing the category system of Wikipedia starting with a set of 132 manually annotated seed categories. As the method relies only on WP's structure any classification task that can be represented by WP categories can be approached this way for any language available in WP. Second, the categories of these classified articles were used to create NE type vectors to classify yet unlabelled articles by computing the similarities between the vectors and unclassified articles' categories. This was done via bootstrapping in two setups that work with two similarity measures: cosine similarity and Dice's coefficient. The results were evaluated on manually annotated data and showed that the type vectors created from the initialization step easily outperform a random baseline and that the method is suited well for the NE types used in MUC-6 (Grishman and Sundheim, 1996) but that the additional CoNLL class MISC shows a gap in quality because it is harder to map the latter to Wikipedia categories. The evaluation of bootstrapping iterations reveals that Dice's coefficient is the better similarity measure for this particular task. This can be attributed to its property of taking the weights of the vectors' values into account in contrast to cosine's property of only observing the angle between two vectors ignoring their lengths. After all, two lists of NEs were created for each of the types PER, LOC, ORG and MISC, one by cosine and one by Dice similarity. Adding NE types to HeiNER makes it a valuable resource for multilingual NERC providing a fair amount of training material in various languages.

6 Acknowledgements

Thanks to Anette Frank for her suggestions and support for the thesis that is the basis of this paper.

References

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 148–151, Morristown, NJ, USA. Association for Computational Linguistics.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy*, pages 9–16, April.

Christian Federmann and Sabine Hunsicker. 2011. Stochastic parse tree selection for an existing rbmt system. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 351–357, Edinburgh, Scotland, July. Association for Computational Linguistics.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 466–471. http://acl.ldc.upenn.edu/C/C96/C96-1079.pdf.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In *Proceedings of the 7th Conference on Natural language Learning at HLT-NAACL 2003*, pages 142–147, Morristown, NJ, USA.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of Conference on Natural Language Learning*.

G. Szarvas, R. Farkas, A. Kocsor, et al. 2006. A multilingual named entity recognition system using boosting and c4. 5 decision tree learning algorithms. *Lecture Notes in Computer Science*, 4265:267.

Wolodja Wentland, Johannes Knopp, Carina Silberer, and Matthias Hartung. 2008. Building a multilingual lexical resource for named entity disambiguation, translation and transliteration. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may.