Entity Classification by Bag of Wikipedia Articles

Tomáš Kliegr
Department of Information and Knowledge Engineering
Faculty of Informatics and Statistics, University of Economics, Prague
tomas.kliegr@vse.cz

ABSTRACT
The input for a Bag-of-Articles (BOA) classifier is a set of unlabeled entities - noun chunks and a set of target labeled entities - Wikipedia articles. The classifier locates Wikipedia articles that might define the unlabeled entity and performs disambiguation selecting one. Both unlabeled and labeled entity is represented with the proposed BOA term weight vector, which is created by aggregating term weight vectors of articles related to the Wikipedia article defining it. The label is assigned by choosing the closest labeled entity, also a BOA term weight vector, with cosine similarity. The paper formally defines the BOA entity representation and BOA-based entity classification and presents a partial software implementation. A BOA-based disambiguation algorithm is presented as a planned extension.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing-Dictionaries; I.2.6 [Artificial Intelligence]: Learning

General Terms
Algorithms, Experimentation

Keywords
Wikipedia, learning, information retrieval, named entity recognition, disambiguation

Introduction
This paper addresses the problem of unsupervised classification of entities represented by noun chunks into an arbitrary set of classes. The noun-chunks come from short textual fragments and thus generally do not have left and right context. This task arises for example in the analysis of image captions. While techniques for Named Entity Recognition and classification (NER) are well-researched, they are not applicable in this setting as NER classifiers typically need to be trained on large labeled document corpora, which generally involve only four labels. In our previous research [9], we have approached this task by mapping the noun chunks as well as classes to Wordnet synsets and then using a graph similarity measure Lin to assign the closest class to the noun chunk. If a noun chunk did not match any Wordnet synset, a Wikipedia article defining the noun chunk was retrieved using a hybrid popularity and text-based relevance metric and a hypernym from this article was extracted using a rule-based extraction grammar. This hypernym was then used to map the noun phrase to a Wordnet synset. While the Wordnet-based classification did not perform well with 52 % accuracy, mapping noun chunks to Wikipedia articles and hypernym extraction showed surprisingly good results, the correct hypernym was extracted for 75 % of named entities not found in Wordnet.

The approach presented here builds upon the encouraging result obtained on Wikipedia and proposes to use it also for classification. Unlike Wordnet, where most information about similarity of two synsets is contained in the path in the thesaurus connecting them while the synset definitions are very short, Wikipedia is less rigidly organized, but contains in average about 300 words per article. By representing the noun chunk and the class as Wikipedia articles, then the noun chunk classification translates to the task of measuring semantic relatedness of two documents.

The main contribution of this paper is the proposal for the Bag-of-Articles (BOA) representation of a given Wikipedia article. BOA of article a is a term-weight vector created by aggregating the term-weight vector of article a with term-weight vectors of articles related to a. The hypothesis is that the BOA representation will yield results comparable to the state-of-the-art algorithms in the task of assessing relatedness of two Wikipedia articles.

This paper presents a formal model and software implementation for experimentation with the BOA representation. This will be applied to identify which measures of relatedness and aggregation techniques are most suitable for the unsupervised entity classification task. The second goal

1Typically person, location, organization and miscellaneous.
of this work is to evaluate the benefit of BOAs for entity disambiguation.

The paper organization is as follows. Section 1 introduces a formal model for entity classification with the BOA representation. A proposal for an entity disambiguation algorithm using BOA is presented in Section 2. Section 3 describes the BOA experimentation system and Section 4 demonstrates its functionality on a simple showcase classification example. Section 5 discusses related work. The conclusions highlight the challenges and suggest possible ways for improvement.

1. BAG-OF-ARTICLES

The proposed Bag-of-Articles (BOA) is an extension of the well-known Bag-of-Words (BOW) approach. The input for a BOA classifier is the classified entity represented as a noun chunk and a set of class entities, represented with a Wikipedia page title. For unlabeled entities, the BOA classifier locates articles in Wikipedia that might define the entity and selects one of them using a disambiguation function. Subsequently, it uses link analysis to try to identify related articles falling into the same semantic category, and then creates a BOA term weight vector by aggregating their BOWs vectors. The class is assigned by choosing the closest class entity, also a BOA term weight vector, with cosine similarity or other suitable metric.

Formally, the input of a BOA classifier is a set of \( l \) labeled instances (titles of Wikipedia articles) \( C \) and a set of \( u \) unlabeled instances (noun phrases) \( E \). Wikipedia article titles provide an unanimous mapping between the labeled instance and a Wikipedia article. We use symbol \( W \) to denote a collection of all pages in Wikipedia at a given time. Each article is described by its title, term weight vector, outbound links, a list of categories it belongs to and type (article page, disambiguation page, category page,...). The BOA representation, as proposed here, does not process Wikipedia infoboxes.

For an unlabeled instance \( e_x \in E \), it is first necessary to determine the articles that may define its various senses. The ranking function \( \rho \) maps it onto the vector of its \( n \) possible senses \( s_x = \rho(e_x,W) = (s_{x,1} \ldots s_{x,1} \ldots s_{x,n}) \). The senses - titles of Wikipedia article pages - are sorted in the vector in the decreasing order of relevance.

The sense \( l \) of an unlabeled instance \( e_x \) is represented by article title \( s_{x,l} \). The fact that there are multiple senses for the unlabeled instance gives space for disambiguation function \( \delta \). In the base scenario, we use disambiguation function \( \delta_{mf,s} \), which assigns the most frequent sense:

\[
\delta_{mf,s}(s_x) = s_{x,1}.
\]

Now, both a disambiguated unlabeled instance and a labeled instance is a Wikipedia article title \( l \) and can be mapped to a Wikipedia article \( a_i \). In the following, we will use the variable \( a \) to refer to a Wikipedia article to which an instance (labeled or unlabeled) is mapped. The bag of articles \( \beta(a) \) is constructed by aggregating related article across the set of modalities \( M \) with the help of the modality membership function \( \mu \), article term-weighting function \( \tau \) and recursive term-weight aggregation function \( \theta \).

Modality membership \( \mu \)

Modality membership function \( \mu(a,a_r) \rightarrow \{0,1\} \) expresses if article \( a_r \) is considered related to \( a \) (\( \mu = 1 \)) or not (\( \mu = 0 \)). Several modality membership functions are suggested below.

Article \( a \) is evaluated as related to \( a_r \) (\( a \neq a_r \)) if:

- \( \mu_{outlink}(a,a_r) = 1 \) iff \( a \) links to \( a_r \),
- \( \mu_{backlink}(a,a_r) = 1 \) iff \( a_r \) links to \( a \),
- \( \mu_{related outlink}(a,a_r) = 1 \) iff \( a \) links to \( a_r \) and there is an article \( a_c \) linking to \( a \) and \( a_r \) and \( a_c \neq a \neq a_r \).
- \( \mu_{firstpara outlink}(a,a_r) = 1 \) iff \( a \) links to \( a_r \) and the link from \( a \) to \( a_r \) is contained in the first paragraph of \( a \)
- \( \mu_{shared category outlink}(a,a_r) = 1 \) iff \( a \) links to \( a_r \) and \( a \) and \( a_r \) share the same category.

Other modality membership function definitions are also possible and various have been in fact suggested in the literature, albeit under a different name. This applies e.g. to \( \mu_{backlinking outlink} \) and \( \mu_{firstpara outlink} \) which is used in the Lucene Search Mediawiki Extension (refer to Section 3).

We use the symbol \( A^m_{a} \) to denote the set of all articles \( a_r \) that are related to \( a \) with respect to modality membership function \( \mu_m \):

\[
A^m_{a} = \{ a_r | a_r \in W, \mu_m(a,a_r) = 1 \}.
\]

The bag-of-articles might contain articles related according to multiple modalities.

Article term-weighting \( \tau \)

The weight function \( \tau(a) \rightarrow R^n \) represents the article \( a \) as a vector of term weights. The parameter \( w_{m,d} \) is a weight assigned to term vectors \( \tau(a) \) in modality \( m \) and depth \( d \). The term weight functions considered are:

- term frequency (TF),
- term frequency * inverse document frequency (TF-IDF) computed over entire Wikipedia,
- term frequency * inverse document frequency computed over articles included in bag of articles of labeled instances \( C \),
- term frequency with first paragraph boost.

Other term-weight function definitions can be also considered.

Recursive term-weight aggregation \( \theta \)

The function \( \theta_m(a,d,maxd_m) \rightarrow R^n \) recursively aggregates term-weight vectors of articles related to \( a \) according to the modality membership function \( \mu_m \):

\[
\theta_m = \begin{cases} 
\sum_{a_r \in A^m_{a}} [w_{m,d} \tau(a_r)] & \text{if } d < maxd_m \\
\theta_m(a,d + 1, maxd_m) & \text{if } d = maxd_m
\end{cases}
\]

\( ^3 \)The first paragraph of a Wikipedia article contains usually the definition of the article subject, it can be therefore expected to contain more relevant words than the rest of the text.
**Bag of articles** \( \beta \)

Function \( \beta(a) \rightarrow R^+ \) creates the bag of articles for article \( a \):

\[
\beta(a) = \tau(a) + \sum_{m \in M} \theta_m(a, 1, \max_d). \tag{4}
\]

The formula aggregates the term-weight vector for article \( a \) with term-weight vectors of articles recursively related to it up to level \( \max_d \). \( \max_d \in N \). The articles (directly) related to it have level 1.

The classification is done by comparing the BOA vector of the unlabeled instance \( \beta(a_x) \) with BOA term vectors of labeled instances \( \beta(a_e) \) with the similarity metrics \( \sim \) and selecting the class with the highest similarity:

\[
\text{BOAclass}(a_x) = \arg\max_{c \in C} \sim(\beta(a_x), \beta(a_e)). \tag{5}
\]

A BOA classifier implementation needs to make decisions as of the selection of the ranking function \( \rho \), modality membership functions \( \mu_m \), term weighting function \( \tau \) and the BOA similarity function \( \sim \). The weights \( w_m,d \) and the maximum depth \( \max_d \) for gathering related pages in modality \( m \) are externally set. The modality functions and weights, the term weighting function and particularly the maximum depth can be set separately for labeled and unlabeled instances.

2. DISAMBIGUATION

This section describes the use of BOA representation for mapping unlabeled entities onto Wikipedia articles. The baseline disambiguation function \( \delta_{m,j} \) described in Section 1 takes the most-frequent sense (MFS) approach, selecting the article with the highest rank assigned by the ranking function to each unlabeled entity; each entity being ranked independently. While the motivational problem for the BOA entity classifier stated in the introduction asserts that the left and right context for an entity is not available, it can be in many classification scenarios safely assumed that all the unlabeled entities come from the same dataset and thus share the same context in general.

For example, consider noun chunks extracted from captions of the Israeli images dataset [1]. This dataset contains 1823 image-captions mostly thematically related to Israel. After the analysis of the captions with entity extraction system [9], the result includes among others the following noun chunks/entities: \( e_1 \) : *The Church Of St Joseph*, \( e_2 \) : *Jerusalem* and \( e_3 \) : *Wadi Qelt*. Considering the first entity \( e_1 \), the first sense \( s_{1,1} = \delta_{m,j}(p(c_1,W)) \) is a Wikipedia article *St. Joseph’s Church, Nazareth*. With entity \( e_2 \) and \( e_3 \), the most frequent sense assumption works well and \( s_{2,1} \) and \( s_{3,1} \) are correct.

The idea for the proposed heuristic disambiguation algorithm is to replace the outlying senses with a sense that better fits with other selected senses. The algorithm contains a clustering step to reflect the fact that the dataset may contain several distinct types of entities, some of which may not share a common context. For example, in the Israeli dataset there are several such entities such as *Rain Pool* or *Apple*. Trying to adjust the sense of these outlying entities with the rest of the collection could result in misclassification.

The algorithm is presented at Listing 1. In each iteration, the algorithm identifies the entity \( s_i \) that is farthest from any of the context clusters and tries to find a sense \( s_{i,k} \) that would better match any of the context clusters than the currently assigned sense \( s_{i,j} \). If no better matching sense is found, the entity is removed from the DIS (to DISambiguate) list and put into the RES (RESolved) list. If a better matching sense is found, it is first checked if the swap \( s_{i,j} \Rightarrow s_{i,k} \) has not already occurred in a previous iteration. If this is the case, the entity is removed from DIS and put into the RES list, otherwise the sense \( s_{i,k} \) replaces \( s_{i,j} \) in DIS and this operation is recorded into the SWAPLOG, a change log of swaps in senses. The algorithm stops either when the DIS list empty, or when the similarity between the most remote entity and its closest cluster exceeds the MAXSIM parameter or when the maximum number of iterations MAXIT is reached. After the disambiguation is performed, the classification is done in the way described in the previous section.

**Listing 1: Heuristic disambiguation algorithm**

```
// possible senses of entity 1...u
S := \{ s_1, ..., s_u \}, where s_i = \{ s_{i,1}, ..., s_{i,n} \}
MAXIT := N
MINCDIS := R

// init DIS with most frequent sense
for each s_i in S
{ append s_i,1 to DIS }
CLUS = cluster(S)

while (it < MAXIT and DIS not empty)
{
    // Identify the biggest outlier
    i,j := arg min_{i,j} \sim(s_{i,j}, c_i) \in CLUS, s_{i,j} \in DIS
    // finish if it is not far enough
    if \sim(s_{i,j}, c_i) > MAXSIM
        break
    
    // Try to get a closer sense
    k := arg max_k \sim(s_{i,k}, c_i) \in CLUS, s_{i,k} \in s_i in S
    // move to resolved if none found
    if j == k
        remove s_{i,j} from DIS, put s_{i,j} to RES
    // move to resolved if there is a cycle
    else if "s_{i,j} \Rightarrow s_{i,k}\" in SWAPLOG
        remove s_{i,j} from DIS, put s_{i,j} to RES
    // otherwise swap with a closer sense
    else
        replace s_{i,j} with s_{i,k} in DIS
        put "s_{i,j} \Rightarrow s_{i,k}\" to SWAPLOG
        update CLUS
    it = it + 1
}
return DIS ∪ RES
```

It should be noted that that Listing 1 presents a first consideration for a disambiguation algorithm, which was not yet implemented. The results obtained by this algorithm may be well-below the most frequent sense baseline, since it
is a widely acknowledged fact that it is difficult even for a supervised system to meet this baseline. To illustrate this, [12] states that “unsupervised systems were found to never outperform the most frequent sense (MFS) baseline (a sense assignment made on the basis of the most frequent sense in an annotated corpus), while supervised systems occasionally perform better than the MFS baseline, though rarely by more than 5%”.

3. IMPLEMENTATION

This section describes an experimental implementation of the BOA-based classification system. As the ranking function $\rho$, the implementation uses a composite metric, which combines text-based similarity between the noun chunk and article text and article popularity as measured by the number of backlinks. As modality membership function $\mu_{m}$, there is one option - outlinks, implementation of backlinks is in progress. For the term weighting function $\tau$, there is a TF and TF-IDF support. As the BOA similarity metrics $sim$, the implementation uses cosine similarity.

A BOA classifier requires a Wikipedia index containing the following pieces of information about each article:

- term vectors with term frequencies,
- outlinks,
- popularity ranking (for most frequent sense relevance ranking)

Given the current size of English Wikipedia and the fact that it is constantly updated, meeting these data acquisition requirements results in a considerable engineering effort and in fact a reimplementation of an existing software as these functions are from the most part performed by the existing Lucene-Search Mediawiki Extension. This Lucene-based Mediawiki search engine indexes the Mediawiki article database and creates several Lucene indexes: main index, headlines index, links index, related index and spellcheck index. For the BOA classifier, the main index containing term vectors and the links index containing links leading out of each article are the most important. This extension provides two additional vital functions for the BOA classifier - parsing of wikitext and prospectively the ability to perform incremental updates.

The main wiki index contains the following important fields: title, key with a numeric article identifier, the term vectors are saved in the contents field, category stores article’s categories, related stores titles of articles that were determined as related during indexing.$^6$

The wiki.links index has the following fields: article key containing concatenated article title, article pageid with a unique numeric identifier that binds the entry with the main index key field, links with a list of article titles to which the article links. The index differentiates between different types of links (article/image) using a namespace (prefix), redirect contains the title of the article to which the current article is redirected, rank contains the number of backlinking articles.

In the BOA classifier implementation, these indexes are exploited as follows.

- **Term vectors** Indexed Wikipedia articles are stored in the main wiki index, however the Lucene-Search extension does not store term vectors. For the purpose of the BOA classifier, it was necessary to modify the extension with code for storing the term vectors.
- **Outlinks** This information can be obtained from the links field of the article entry in the wiki.links index.
- **Popularity ranking** The Lucene-Search Extension contains a search engine, which uses sophisticated relevance ranking involving the number of backlinks. The BOA implementation uses the first-ranked article as the MFS baseline. The Lucene Mediawiki Indexer as used in the BOA classifier system has several changes in code, the most marked one is the extension of the index with stored term vectors. The term vector computations are done with a sparse matrix toolkit java library.$^7$

4. EXPERIMENTAL EVALUATION

The implementation as of submitting the article reached the state that it allows first preliminary evaluation.

To assess the performance of the implementation in terms of speed, we have used the system to determine whether the noun chunk “Queen Mary, University of London” is a building or person.

The implementation was initialized with a dump of the English Wikipedia from March 2010. The dump was restored and indexed by the modified indexer from the Lucene Mediawiki Search Extension, which created the main wiki index (17.6 GB, 5.5 million documents) and the wiki.links index (5.9 GB, 9.6 million documents).

The BOA representations for training were created with the following setting: $m = \{\text{outlink}\}, \maxd_{\text{outlink}} = 3$, and for testing $m = \{\text{outlink}\}, \maxd_{\text{outlink}} = 1$. Disambiguation is done with the most frequent sense assumption and TF-IDF for the weighting function $\tau$. The training took 84 seconds on an Intel Core i3 laptop. The BOA representation resulted into a term-weight vector of length 29,316, the Person class had 19,816 nonzero weights and the Building class 16,276. The classifier correctly assigned the Building class, with the difference in similarities being 0.029.

The results presented above are primarily intended to demonstrate the functionality of the implementation. However, the small gap between the differences in similarities indicates unsuitability of the outlink modality. This is consistent with intuition, as just following links leading out of an article will include many semantically unrelated articles. It remains to be seen of this problem is alleviated by using the related outlink or shared category outlink modalities. These could increase the odds that the included articles share the same context.

Another problem is the fact that evaluating a single test instance is very slow, due to many costly reads from the Lucene index. This may not be an issue for experiments on smaller testsets, but for real-world datasets a more efficient implementation is necessary. Speed optimization may be achieved by keeping the entire index in memory or using the Lucene’s Machine Learning subproject Apache Mahout.$^8$

For a more thorough analysis, a publicly available dataset

---

$^4$http://www.mediawiki.org/wiki/Extension:Lucene-search

$^5$http://lucene.apache.org

$^6$A is said to be related to B, if A links to B, and there is some C that links to both A and B (source: Lucene-Search Extension documentation).

$^7$http://code.google.com/p/matrix-toolkits-java/

$^8$http://mahout.apache.org
will be used. There are established test collections for the related Named Entity Recognition tasks. These collections are in principal applicable, however most dataset interpret the NER task as classification into four general classes (PERSON, LOCATION, ORGANIZATION, MISCELLANEOUS). Performance of a BOA implementation on such a general set of classes has little generalization potential for use with datasets involving many specific target classes. Since labels are not required by the proposed algorithm for training, a dataset for testing and parameter tuning is required. It might be feasible to create such a dataset by annotating entities in an already existing one, such as the Israeli images dataset [1].

The most suitable candidate for a test dataset seems to be however the WordSimilarity-353 collection [5] containing 353 word pairs human annotated for semantic relatedness. Some results achieved on this dataset are discussed in the following section.

5. RELATED RESEARCH

There is a quickly increasing body of research exploiting Wikipedia for various NLP tasks. Paper [8] explores the use of Wikipedia as external knowledge to improve named entity recognition (NER). Their method retrieves the corresponding Wikipedia entry for each candidate word sequence by comparing the word sequence with Wikipedia article title. The candidate word sequences are all sequences of no more than eight words that start with a word containing at least one capitalized letter. Then they perform POS tagging and noun phrase chunking on the first sentence of the article and extract the last word from the first noun phrase after the first "is", "was", "are", or "were" in the sentence. This noun can be considered as a hypernym for the candidate and is used as a feature in a CRF-based named entity tagger. Using the CoNLL 2003 dataset, the authors report that out of 256,418 search candidates, for 48,845 a matching article was found. This can be used as a gross estimate of the number of cases when Wikipedia-based classification can be applied.

Wikipedia was used in a similar way in our earlier paper [9] with some differences: as candidate word sequences we used noun phrases, the search for a corresponding Wikipedia article involved also the article’s popularity ranking and text, and the extracted hypernym is used in a Wordnet-based classifier not a CRF tagger.

One of the closest approaches to the BOA entity classification proposed in this paper constitute the systems WikiRelate! [13] and Explicit Semantic Analysis (ESA) [6].

Given a pair of words $w_1$ and $w_2$, WikiRelate! searches for Wikipedia articles $p_1$ and $p_2$ that respectively contain $w_1$ and $w_2$ in their titles. Semantic relatedness is then determined by graph based measures computed over a taxonomy created from article categories and by content-based measures computed over texts of $p_1$ and $p_2$.

In the ESA method, the input text $T$ is represented as a TF-IDF term vector. For each word $w_i$ in the input text the method uses an inverted index to retrieve Wikipedia articles $c_1 \ldots c_n$ containing $w_i$. The semantic relatedness of the word $w_i$ with concept $c_j$ is computed such that the strength of association between $w_i$ and $c_j$ is multiplied with the TF-IDF weight of $w_i$ in $T$. The relatedness score for any two documents is determined by computing the cosine similarity between the vectors of document-concept semantic relatedness. The ESA method was evaluated on two tasks - individual word relatedness and document relatedness. The former task is of high relevance to the entity classification task, therefore we report the results in detail. The result for ESA is 0.75 correlation with humans, which as the paper reports surpasses previous results including Wordnet [7] (0.33-0.35), Latent Semantic Analysis [5] (0.55) and WikiRelate! (0.19-0.48). The evaluation dataset for all experiments was the WordSimilarity-353 collection [5].

ESA is more similar to the BOA entity classifier than WikiRelate, because it represents an entity by multiple articles from Wikipedia that are determined as relevant to it. However, the difference between BOA and ESA is in the way how these additional pages are identified: ESA searches the word in inverted index disregarding the hyperlink structure while BOA finds a seed page and then examines its neighborhood, possibly taking advantage of the category information as used in WikiRelate!.

Another difference is in the way the semantic relatedness is assessed, while both methods use cosine similarity, BOA uses the full term frequency vectors, while ESA uses shorter document-concept semantic relatedness vectors. While the shorter term vectors indisputably favour fast evaluation, it remains to be seen whether the longer term-frequency vectors will compensate slower performance with improved results. It should be noted that the higher dimensionality can also have an opposite effect due to the curse of dimensionality phenomenon. The highest benefit can be however expected from exploiting the hyperlink structure. It was shown in [11] that relying purely on the hyperlink structure and disregarding its category hierarchy and textual content, it is possible to achieve 0.69 correlation on WordSimilarity-353. This result underperforms ESA only by a thin margin, while using much less information.

There is also a large body of work on entity disambiguation, however only a limited number of algorithms exploit Wikipedia as its primary resource. One of the most relevant of such approaches is presented in the paper entitled System for large-scale named entity disambiguation based on Wikipedia [3]. Below, we compare it to the disambiguation algorithm proposed in Section 2.

The input text is analyzed using NLP methods and with the help of web search query logs for occurrence of entities to be disambiguated. These are called surface-forms, while we call them noun chunks. The system then tries to map each surface form to the corresponding entity article, which is a Wikipedia article focused on a single entity.

The Wikipedia entity articles that have a surface form matching the disambiguated surface form are considered as candidates. Entity surface forms are created from the titles of entity pages, titles of redirecting pages and the references to entity pages in other Wikipedia articles. In our approach, we match candidate articles by full-text search in Wikipedia, which uses a number of backlinks as one of the parameters. The advantage of this approach is that it ranks first the (likely) most frequent of the candidate entities.

For each candidate, an entity vector of contexts and categories is created. The context component contains articles to which the candidate entity article links from the first paragraph and articles for which the corresponding pages refer back to the candidate entity. The categories component contains category tags that are either directly assigned to the article or extracted from Wikipedia list pages it appears on.

The entity vector for a candidate entity is a 0-1 vector...
of length M+N, where M is the number of known contexts and N the number of known category tags. The role of the entity vector is similar to that of the BOA vector. However, the elements of the BOA vector are term weights, while the elements of the entity vector are binary values – 1 if category/context does appear for the candidate and 0 otherwise.

The entity vector is to be considered as a disambiguated document, because it reflects the authors experiments with broader inclusion strategy (e.g. all linked articles), which yielded worse result. In contrast, the recursive definition of BOA includes also articles that are not directly linked with the candidate entity. The argument is that due to the use of the term-weight vectors, an article might contribute a valuable information to the bag even if only its small fragment contains words that also occur in the same context in another article that is in a BOA representation of another disambiguated entity. The effect of frequently occurring words share by many articles can be mitigated by the use of IDF in the term weighting function. In the entity vector approach, including an article is beneficial primarily if the article itself is found in the context of another disambiguated entity.

The disambiguation is performed in the following way. A disambiguated document is represented with a document vector, which aggregates all possible entity disambiguations (their contexts and categories) of each surface form appearing in the document. This document vector is subsequently compared with entity vector of each possible entity disambiguation and the assignment of entities to surface forms that maximizes the similarity of the vectors is selected.

We suggest to use a more subtle representation against which candidate entities are compared to than the aggregate of all possible senses in the document vector is. In our approach, the candidate entity is compared with several clusters that are supposed to group entities of like meaning. The number of necessary comparisons is alleviated by the initialization of sense assignment to the most frequent sense and by attempting to change the sense only for entities that do not fit well any cluster.

6. CONCLUSION

There is an increasing number of evidence showing that the Wikipedia Encylopedia can be a valuable dataset for NLP and information retrieval tasks.

The commonly used datasets used for training and evaluation of NLP algorithms such as Brown corpus, the 20 newsgroups or Senseval competition datasets consist of annotated unconnected text files. In contrast, Wikipedia documents are structured, interlinked and organized into a taxonomical system of categories. Wikipedia thus poses a challenge for development of new algorithms that take advantage of all these pieces of information, some of which is unique to Wikipedia and cannot be found in a generic web document. Consider for example the categories or the opening definition, which often contains hypernym for the entity described by the article [10].

This paper presented an ongoing work on the problem of unsupervised entity classification using Wikipedia. The Bag-of-Articles representation was specifically proposed so that it exploits some of the features Wikipedia offers over the generic plain text corpora. An implementation of an unsupervised entity classification system using this representation has been also described. The system has heuristically chosen component functions for article ranking, modality membership, term weighting and the BOA similarity and parameters. While experiments with this implementation will bring insight into the value of this approach, obtaining performance competitive to the state-of-the-art ESA algorithm [6] is a challenging task, which will likely require devising a method for automatic parameter and component adjustment, either using a labeled training dataset or using cross-validation on an evaluational dataset.

Acknowledgment. The research was partially supported by the European Commission under contract FP7-216444 PetaMedia, and by the Ministry of Education, Youth and Sports, of the Czech Republic under grant No. ME913, and by grant No. 201/08/0802 of the Czech Science Foundation.

7. REFERENCES


