# Scalable Disambiguation System Capturing Individualities of Mentions

Tiep Mai[1], Bichen Shi[2](✉), Patrick K. Nicholson[1], Deepak Ajwani[1], and Alessandra Sala[1]

[1] Nokia Bell Labs, Dublin, Ireland
{tiep.mai,patrick.nicholson, deepak.ajwani,alessandra.sala}@nokia-bell-labs.com
[2] University College Dublin, Dublin, Ireland
bichen.shi@insight-centre.org

**Abstract.** Entity disambiguation, or mapping a phrase to its canonical representation in a knowledge base, is a fundamental step in many natural language processing applications. Existing techniques based on global ranking models fail to capture the individual peculiarities of the words and hence, struggle to meet the accuracy-time requirements of many real-world applications. In this paper, we propose a new system that learns specialized features and models for disambiguating each ambiguous phrase in the English language. We train and validate the hundreds of thousands of learning models for this purpose using a Wikipedia hyperlink dataset with more than 170 million labelled annotations. The computationally intensive training required for this approach can be distributed over a cluster. In addition, our approach supports fast queries, efficient updates and its accuracy compares favorably with respect to other state-of-the-art disambiguation systems.

**Keywords:** Entity linking · Entity disambiguation · Wikification · Word-sense disambiguation

## 1 Introduction

Many fundamental problems in natural language processing, such as text understanding, automatic summarization, semantic search, machine translation and linking information from heterogeneous sources, rely on entity disambiguation [6,22]. The goal of entity disambiguation and more generally, word-sense disambiguation is to map potentially ambiguous words and phrases in the text to their canonical representation in an external knowledge base (e.g., Wikipedia, Freebase entries). This involves resolving the word ambiguities inherent to natural language, such as homonymy (phrases with multiple meanings) and synonymy (different phrases with similar meanings), thereby, revealing the underlying semantics of the text.

T. Mai—Now at TrustingSocial (tiep@trustingsocial.com).

J. Gracia et al. (Eds.): LDK 2017, LNAI 10318, pp. 365–379, 2017.
DOI: 10.1007/978-3-319-59888-8_31

**Challenges:** This problem has been well-studied for well over a decade and has seen significant advances. However, existing disambiguation approaches still struggle to achieve the required *accuracy-time trade-off* for supporting real-world applications, particularly those that involve streaming text such as tweets, chats, emails, blogs and news articles.

A major reason behind the *accuracy* limitations of the existing approaches is that they rely on a single global ranking model (unsupervised or supervised) to map all entities. In a sense, such inflexible methods use a single rule set (a single trained/unsupervised model) for the disambiguation of all text phrases. Apart from their meanings, the phrases also differ in their origins, emotional images they evoke, their general popularity, their usage by demographic groups as well as in how they relate to the local culture. Hence, even synonymous phrases can have very different probability distribution of being mapped to different nodes in the knowledge base. However, global ranking models do not customize disambiguation rules per text phrase, fail to capture the subtle nuances of individual words and phrases in the language, and are, thus, more prone to mistakes in entity disambiguation.

Some systems perform joint disambiguation on multiple text phrases together for accuracy improvement. However, due to the utilization of pairwise word-entity, entity-entity interactions or even combinatorial interactions, many joint disambiguation approaches suffer from slow *query time*.

**Our Approach:** We propose a novel approach to address all of these issues in word-sense disambiguation. Our approach aims at learning the individual peculiarities of entities (words and phrases) in the English language and learns a specialized classifier for each ambiguous phrase. This allows us to find and leverage features that best differentiate the different meanings of each phrase.

To train the hundreds of thousands of classifiers for this purpose, we use the publicly available Wikipedia hyperlink dataset. This dataset contains about 170 million annotations. Since training each classifier is an independent task, our approach can be easily parallelized and we use a distributed Spark cluster for this purpose. The small number of features used in these classifiers are based on text overlap and are, therefore, light-weight enough for its usage in real-time systems. We consider this parallelization to be an important advantage of our approach of learning specialized and independent classifier for each mention (as most global supervised and unsupervised approaches are non-trivial to parallelize, if they can be parallelized at all).

Updating our system for new entities (e.g.,"Ebola crisis", "Panama papers", "Migrant crisis") as well as for changing meanings of existing entities (e.g., the phrase "US President" has a higher prior of referring to "Donald Trump" after Jan. 20, 2017 and to "Barack Obama" for the previous eight years) simply requires learning the models for those entities, and does not affect the other classifiers. In contrast, existing state-of-the-art approaches would either fail to capture such changes in semantics of individual entities or require significant amount of time to update their global models.

Furthermore, unlike the increasingly popular deep learning architectures, our approach is interpretable: it is easy to understand why our models chose a particular mapping for a phrase.

We provide an extensive experimental evaluation to show that even though our system was designed to support fast disambiguation queries (average less than 3 ms) and enable efficient updates, the accuracy of our approach is comparable to many state-of-the-art disambiguation systems.

**Outline:** The rest of the paper is organized as follows. Section 2 presents related disambiguation techniques. Section 3 gives an overview of the Wikipedia hyperlink data used in the training of our disambiguation system. In Sect. 4, we present the details of our novel disambiguation approach. Sections 5, 6 and 7 present the experimental results of comparing with other disambiguation systems, using both Wikipedia data and the benchmark framework GERBIL [24].

## 2 Related Work

There is a substantial body of work focussing on the task of disambiguating entities to Wikipedia entries. The existing techniques can be roughly categorized into unsupervised approaches that are mostly graph-based and supervised approaches that learn a global ranking model for disambiguating all entities.

**Graph-Based Approaches:** In these approaches, a weighted graph is generally constructed with two types of nodes: phrases (mentions) from the text and the candidate entries (senses) for that phrase. For the mention-sense edges, the weights represent the likelihood of the sense for the mention in the text context. For the sense-sense edges, the weights capture their relatedness, e.g. the similarity between two Wikipedia articles in terms of categories, in-links, out-links. A scoring function is designed and then optimized on the target document so that a single sense is associated with one mention. Depending on the scoring function, this optimization can be solved using one of the following algorithms:

- Densest subgraph algorithms on an appropriately defined semantic graph and selecting the candidate sense with maximum score [11,18]
- Random walk techniques and choosing the candidate senses by the final state probability [8,10]
- Some path-based metrics for joint disambiguation [13]
- A centrality measure based on HITS algorithm on a DBpedia subgraph containing all the candidate senses (AGDISTIS approach) [23]
- PageRank on the mention-entity graph where the transition probabilities are evaluated by Word2Vec semantic embeddings and Doc2Vec context embeddings [25]
- Other centrality measures such as variant of Betweenness, Closeness, Eigenvector and Degree centrality [1]
- A probabilistic graphical model that addresses collective entity disambiguation through the loopy belief propagation [7]

Since these graph-based solutions are mostly unsupervised, there is no parameter estimation or training during the design of the scoring function to guarantee the compatibility between the proposed scoring function and the observed errors in any trained data [10,11,20]. Some disambiguation systems do apply a training phase on the final scoring function (e.g., TAGME [5]), but even here, the learning is done with a global binary ranking classifier. An alternative system uses a statistical graphical model where the unknown senses are treated as latent variables of a Markov random field [14]. In this system, the relevance between mentions and senses is modeled by a node potential and trained with max-margin method. The trained potential is combined with a non-trained measure of sense-sense relatedness, to form the final scoring function. However, maximizing this scoring function is NP-hard and computationally intensive [5].

**Supervised Global Ranking Models:** On the other hand, non-graph-based solutions [4,9,15–17,19] are mostly supervised in the linking phase. Milne and Witten [17] assumed that there exists unambiguous mentions associated with a single sense, and evaluated the relatedness between candidate senses and unambiguous mentions (senses). Then, a global ranking classifier is applied on the relatedness and commonness features. Not relying on the assumption of existing unambiguous mentions, Cucerzan [2] constructed document attribute vector as an attribute aggregation of all candidate senses and used scalar product to measure different similarity metrics between document and candidate senses. While the original method selected the best candidate by an unsupervised scoring function, it was later modified to use a global logistic regression model [3].

Han and Sun [9] proposed a generative probabilistic model, using the frequency of mentions and context words given a candidate sense, as independent generative features; this statistical model is also the core module of the public disambiguation service DBpedia Spotlight [4]. Olieman et al. [19] proposed various adjustments (calibrating parameters, preprocessing text input, merging normal and capitalized results) to adapt Spotlight to both short and long texts. They also used a global binary classifier with several similarity metrics to prune off uncertain Spotlight results. Houlsby and Ciaramita [12] employed a probabilistic model based upon Latent Dirichlet Allocation (LDA), and proposed a scalable Gibbs sampling scheme that exploits sparsity in the Wikipedia-LDA model.

In contrast to these approaches that learn a global ranking model for disambiguation, our approach constructs specialized features by contrasting the Wikipedia contexts of candidate senses, and learns a specialized model for each unique mention. This specialization is the main factor that enables our proposed system to achieve high accuracy, fast queries and efficient updates.

**Per-mention Disambiguation:** In terms of per-mention disambiguation learning on the Wikipedia knowledge base, the method by Qureshi et al. [21] is the most similar to our proposed method. However, as their method only uses Wikipedia links and categories for feature design and is trained with a small Twitter annotation dataset (60 mentions), it does not fully leverage the significantly larger Wikipedia annotation data to obtain highly accurate per-mention trained models. Also, while our feature extraction procedure is light and tuned to contrast different candidate senses per mention, their method extracts related categories, sub-categories and articles up to two depth level for each candidate sense, and requires pairwise relatedness scores between candidate sense and context senses. All these high cost features are computed on-the-fly due to the dependency on the context, potentially slowing down the disambiguation process.

## 3 Annotation Data and Disambiguation Problem

We begin with an example to illustrate terminology. Consider the sentence, "Java is a language understood by my computer," and focus on the underlined phrase, "Java". A human can easily link this phrase to its corresponding *entity*, `Java_(programming_language)`, by understanding that the *context* (i.e., the sentence) refers to a programming language. However, this is a non-trivial task, as there are numerous other *senses* of this phrase, such as, `Java_(island)` and `Java_(coffee)`.

Since the senses of phrases are subjective, the first task is to fix a knowledge base and produce a mapping between phrases and senses. For this purpose, we use Wikipedia as our knowledge base.[1] From Wikipedia, we extract the text bodies from Wikipedia entities (i.e., articles) $e$. In each entity's text body, there are hyperlink texts, linking text phrases to other Wikipedia entities. These hyperlink texts are called *annotations*; their associated text phrases and Wikipedia entities are called *mentions* and *senses*, respectively. In terms of the example above, if the example sentence appeared on some Wikipedia page in which the phrase Java was linked to the Wikipedia page `Java_(programming_language)`, we would refer to the combination of the hyperlink and phrase as an annotation: "Java" would be the mention, and `Java_(programming_language)` would be the sense.

We extract all such annotations $a$, linking mentions $m$ to Wikipedia senses $e$[2]. Each annotation includes an annotation context, which is a number of sentences extracted from both sides of the annotation, such that the number of words on each side exceeds a predefined threshold. This threshold is set to 50 in this paper. During the extraction, text elements such as text bodies, mentions, annotation contexts are lemmatized using the python package nltk[3] for the purpose of grouping different forms of the same term. This extracted dataset is denoted by $\mathcal{A}$ in the sequel.

**Formal Problem Statement:** The extracted annotations are grouped by their mentions. For a single unique mention $m$ such as "Java", we obtain the list of distinct candidate senses $E(m)$ from the annotation group of mention $m$, e.g. `Java_(programming_language)`, `Java_(island)`, `Java_(coffee)`. In the *disambiguation problem*, given a new unlinked annotation $a$ with its mention $m$ and context, one wants to find correct destination sense $e$ among all candidate senses $E(m)$.

[1] We used WikiExtractor (http://medialab.di.unipi.it/wiki/Wikipedia_Extractor) on the 2015-07-29 dump.

[2] In our notation, a sense is a Wikipedia entity and is coupled with a specific mention.

[3] http://www.nltk.org/.

## 4 Disambiguation Method

**Disambiguation:** We use a big data approach with supervised discriminative machine learning models for the disambiguation problem. In our approach, all annotations with the same lemmatized mention are grouped together and one multi-class classifier is learnt for each lemmatized mention only using the annotations corresponding to it.

We use the light-weight and robust word-based similarity features between annotation context and sense text body, and show that coupling the specialized per-mention classifier with these features, which are tuned to contrast candidate senses, can deliver a very accurate and fast disambiguation solution. We also tried other more complex features, but they turned out to be either too costly or not as good as similarity features.

For each unique mention $m$, we first construct a local tf-idf matrix for the text bodies of all candidate senses $E(m)$. For each candidate sense $e$ in $E(m)$, we consider the top $n_1$ words, ranked by tf-idf values. We then evaluate the similarity between an annotation context and a candidate sense by measuring the overlap between the set of annotation-context-words and the set of sense-text-body-words.

The overlap metrics are weighted in 4 different ways: *(a)* the overlap between context-words and text-body-words (number of common words in the two sets); *(b)* the overlap weighted by the tf-idf of the sense text body; *(c)* the overlap weighted by the word count of the annotation context; *(d)* the overlap weighted by the product of tf-idf and the word count. For standardization, the metrics are scaled by logarithm of the context length, which can be different for different annotations.

To further improve the accuracy, the $n_1$ words in the annotation context are divided, in order of their tf-idf values, into $n_2$ parts. In the classification model, the various overlap metrics for each part are treated as separate features, thus enabling the different tf-idf value-bands to play different roles in measuring the overall similarity.

We then group all weighted metrics of all candidate senses together as a single feature vector and learn a different multinomial logistic regression model for each mention. The size of the feature vector for a mention $m$ is $4 * length(E(m)) * n_2$. After the learning process, the estimated model can be used to disambiguate new unlinked annotations. The complexity for each disambiguation of unlinked annotations is linear with respect to the context length and the number of candidate senses.

The key point in the above process is the per-mention learning. By doing so, we can leverage the local tf-idf construction among candidate senses to learn highly discriminative words specific to each mention. For instance, for the mention "Java", we can extract words such as "code", "machine", "drink", "delicious", that best discriminate between its different senses like "Java_(programming_language)", "Java_(coffee)". This is different from constructing features from a single global tf-idf of all Wikipedia articles, which suffers from noisy and unrelated Wikipedia articles. Furthermore, this procedure allows flexible weighting of words and features among different unique mentions, capturing the individual nuances of mentions to improve the disambiguation accuracy. The idea of this procedure is analogous to the localization property of kernel method and smoothing spline in machine learning.

**Pruning:** Like other annotation systems, our system has a pruner which can be enabled to remove uncertain annotations and balance the trade-off between precision and recall. However, our pruning is performed on the per-sense level.

The output of the previous multinomial logistic regression model includes both the predicted senses and the probability. Annotations with same predicted sense are grouped together. By comparing the predicted probabilities with the ground-truth, we obtain, for each sense, a list of probability scores for the correct and a list for incorrect annotations. Then, for each sense, we adjust its probability threshold to maximize the precision, subject to the constraint that the F1 should be higher than a predefined value. Thus, for each sense, we get a threshold value specific to it and we use these thresholds to prune at a per-sense level. This procedure can be easily modified to optimize F1-measure or any predefined criteria. Due to the space constraint, the pruning experiments for tuning the constraint of F1-measure and precision are omitted.

## 5 Experimental Set-up

One of the numerical challenges for this approach is the required computation power needed for the processing of more than 700K of unique ambiguous mentions and 170 million labelled annotations. Fortunately, as the feature construction and classification learning is per-mention, the disambiguation system is highly compatible with a data-parallel computation system. So, in order to deal with the numerical computation, we use Apache Spark[4], a distributed processing system based on Map-Reduce framework, for all data processing, feature extraction and model learning. Our Spark cluster consists of three 16 × 2.6 GHz 96 GB-RAM machines. All the algorithms and procedures are implemented in Python with PySpark API. For machine learning methods, we use the standard open source library scikit-learn[5].

**Training and Validation Set-up:** For the purposes of training and validation, the annotation dataset $\mathcal{A}$ in Sect. 3 is split by ratio (90%, 10%) per-mention. The 90% training dataset is denoted by $\mathcal{A}_1$ and the other is by $\mathcal{A}_2$. In order to validate the disambiguation system in different data scenarios such as short-text and noisy-text, we use the following transformation on the original annotation dataset $\mathcal{A}$ and create different validation sets (aside from the original validation set $\mathcal{A}_2$).

For a mention $m$ and its candidate senses, we construct a noisy vocabulary by the unique words of the text bodies of the candidate senses. Then, for every original annotation of $m$ in $\mathcal{A}$, we form a new annotation by sampling a fraction of

[4] http://spark.apache.org/.

[5] http://scikit-learn.org/stable/.

**Table 1.** Data transformation parameters

| Dataset | $\mathcal{B}$ | $\mathcal{C}$ | $\mathcal{D}$ | $\mathcal{E}$ |
|---|---|---|---|---|
| $p_1$ | 80% | 60% | 40% | 20% |
| $p_2$ | 20% | 0% | 0% | 0% |

original context-words with ratio $p_1$, and a fraction of noisy vocabulary with ratio $p_2$. For instance, given $p_1 = 80\%$, $p_2 = 20\%$, the new annotation contains 80% of the original content (randomly sampled) with 20% noisy. Four such datasets are constructed with parameters $p_1$, $p_2$ specified in Table 1 and are only used for validation purpose. We would like to see how the disambiguation system performs in short text environemnt (small values of $p_1$) or in the case where the real context words are contaminated by random context words (non-zero value of $p_2$).

**Metrics:** We use the standard metrics, precision $\mathcal{P}$ and recall $\mathcal{R}$, for evaluating our system. As the above metrics may be biased to mentions with a large number of labelled annotations in Wikipedia dataset, we also use a slightly different precision $\underline{\mathcal{P}}$ and recall $\underline{\mathcal{R}}$, which are averaged by per-mention precision and recall metrics across all mentions.

## 6 Analysis on Learning Settings

In this section, we explore and analyze the accuracy of the proposed disambiguation system.

In the feature extraction step, $n_1$ defines the number of unique words, ranked by tf-idf values, in each candidate sense context, used for matching with an annotation context. In the case of using a large value of $n_1$, we may expect the effect of high ranking words to the disambiguation classifier is different from the ones of low ranking words, and hence divide them in a number of parts $n_2$, as described in Sect. 4. In terms of computation, $n_1$ affects the cost of matching the annotation context with the top-ranked words of candidate context while $n_2$ affects the number of training features.

Another variable that affects the system performance is the classifier. Through preliminary experiments which are omitted from this paper due to the page limit, we find multinomial logistic regression to be the best in terms of accuracy and time complexity for this problem.

For this analysis of configurable system variables, the system is trained and evaluated on 3.4 million random annotations of 8834 randomly selected unique mentions. The validation results are provided for both the original validation dataset $\mathcal{A}_2$ and the scrambled datasets described in Sect. 5

Performance results by varying $n_1$ and $n_2$ with multinomial logistic regression are given in Table 2. The validation on $\mathcal{A}_2$ follows the holdout approach while the other validation results are evaluated on modified test sets (with shrinked contexts and random context words). $\mathcal{T}_{\text{total}}$ is the total time of feature construction, training and validation of all datasets and $\mathcal{T}_{\text{pred}}$ is the prediction time

**Table 2.** Performance results of different settings $(n_1, n_2)$ with multinomial logistic regression. The best results are in bold.

| $n_1$ | $n_2$ | $\mathcal{P}^{\mathcal{A}_2}$ | $\underline{\mathcal{P}}^{\mathcal{A}_2}$ | $\mathcal{P}^{\mathcal{B}}$ | $\underline{\mathcal{P}}^{\mathcal{B}}$ | $\mathcal{P}^{\mathcal{C}}$ | $\underline{\mathcal{P}}^{\mathcal{C}}$ | $\mathcal{P}^{\mathcal{D}}$ | $\underline{\mathcal{P}}^{\mathcal{D}}$ | $\mathcal{P}^{\mathcal{E}}$ | $\underline{\mathcal{P}}^{\mathcal{E}}$ | $\mathcal{T}_{\text{total}}(\times 10^3\text{s})$ | $\mathcal{T}_{\text{pred}}(\text{ms})$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 400 | 8 | **.9186** | .9206 | **.9325** | **.9274** | **.9351** | **.9529** | **.9053** | **.9260** | **.8550** | **.8787** | 47.56 | 5.69 |
| 100 | 2 | .9157 | .9163 | .9243 | .9203 | .9225 | .9347 | .8947 | .9098 | .8487 | .8686 | 2.55 | 3.00 |
| 400 | 1 | .9152 | **.9215** | .9213 | .9186 | .9182 | .9296 | .8951 | .9106 | .8532 | .8754 | 24.32 | 3.91 |
| 100 | 1 | .9138 | .9188 | .9193 | .9163 | .9160 | .9263 | .8916 | .9063 | .8491 | .8701 | **18.13** | **2.81** |

**Table 3.** Results of setting $(n_1 = 100, n_2 = 1)$ for entire Wikipedia

| $\mathcal{P}^{\mathcal{A}_2}$ | $\underline{\mathcal{P}}^{\mathcal{A}_2}$ | $\mathcal{P}^{\mathcal{B}}$ | $\underline{\mathcal{P}}^{\mathcal{B}}$ | $\mathcal{P}^{\mathcal{C}}$ | $\underline{\mathcal{P}}^{\mathcal{C}}$ | $\mathcal{P}^{\mathcal{D}}$ | $\underline{\mathcal{P}}^{\mathcal{D}}$ | $\mathcal{P}^{\mathcal{E}}$ | $\underline{\mathcal{P}}^{\mathcal{E}}$ | $\mathcal{T}_{\text{total}}(\times 10^3\text{s})$ | $\mathcal{T}_{\text{pred}}(\text{ms})$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| .9188 | .9220 | .9261 | .9172 | .9238 | .9265 | .9012 | .9067 | .8617 | .8712 | 1400.77 | 2.82 |

per-annotation (including the feature construction time); both are measured in a sequential manner as the running time of all mentions in all Spark executor instances is summed up before the evaluation.

As we want to validate purely the disambiguation process, we do not prune off uncertain predictions in this section and the disambiguation always returns a non-NIL candidate for any annotation. Consequently, precision, recall and F1-measure are all equivalent and only precision values are reported. We make the following observations about Table 2:

- Increasing $n_1$ and $n_2$ raises the precision but the increment magnitude is diminishing.
- There is a trade off between precision and running time/prediction time. If more top-ranked candidate context words and number of features are considered, the result is higher precision but slower training per-mention/prediction time per-annotation.
- The precision decreases when the context length is reduced between validation datasets $\mathcal{C}$ and $\mathcal{E}$.
- Between dataset $\mathcal{B}$ and $\mathcal{C}$, $\mathcal{B}$ has a longer but noisier context than $\mathcal{C}$, resulting in a lower precision.

The trends are clear without any random fluctuation, indicating experiment stability.

Our last experiment in this section extends to all Wikipedia mentions of more than one candidate senses. Due to the long processing time of more than 170 million annotations, we only run the system with one setting $(n_1 = 100, n_2 = 1)$. The precision results and time statistics are presented in Table 3, and it can be seen that the full performance results are stable and comparable to the ones of the corresponding settings in Table 2.

## 7 Comparison to Other Systems

A big advantage of our system *Per-Mention Learning* (*PML*) is that it has very fast sequential query time (less than 3ms on average). The only other system

**Table 4.** Comparison of DBpedia Spotlight (DS) and our proposed system (PML)

| DS instance ($\gamma$) | $\lvert\mathcal{G}'\rvert$ | $\mathcal{P}_{DS}$ | $\underline{\mathcal{P}}_{DS}$ | $\mathcal{P}_{PML}$ | $\underline{\mathcal{P}}_{PML}$ |
|---|---|---|---|---|---|
| 0.0 | 65k | .8781 | .8169 | .9035 | .8985 |
| 0.5 | 64k | .8822 | .8201 | .9051 | .8989 |

**Table 5.** Comparison of TAGME (TM) and our proposed system (PML)

| $\lvert\mathcal{G}'\rvert$ | $\mathcal{P}_{TM}$ | $\underline{\mathcal{P}}_{TM}$ | $\mathcal{P}_{PML}$ | $\underline{\mathcal{P}}_{PML}$ |
|---|---|---|---|---|
| 37872 | .8752 | .8244 | .9077 | .8950 |

**Table 6.** GERBIL v.1.2.2 comparison of different systems. The micro-F1 (top) and macro-F1 (bottom) scores of each system on each dataset are reported. Each column displays the best micro/macro-F1 score in red (marking the row with †), and the second best micro/macro-F1 score in blue (marking the row with ‡). An archived version of the GERBIL experiment (for all systems except for PML) can be found at http://gerbil.aksw.org/gerbil/experiment?id=201604050003.

| | ACE2004 | AIDA-CoNLL | AQUAINT | DBSpotlight | IITB | KORE50 | Micropost | MSNBC | N3-Reuters-128 | N3-RSS-500 | OKE-2015 | Macro-Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PML | .637<br>**†.793** | **‡.545**<br>**‡.571** | .685<br>.683 | **†.806**<br>**†.812** | .460<br>**‡.459** | .403<br>.376 | .527<br>.729 | .573<br>**†.648** | **‡.553**<br>**‡.592** | **†.677**<br>**†.676** | .737<br>.742 | **‡.600**<br>**†.644** |
| AGDISTIS | .618<br>.752 | .498<br>.491 | .508<br>.495 | .263<br>.273 | **‡.467**<br>**†.480** | .323<br>.290 | .323<br>.593 | **‡.621**<br>.569 | **†.642**<br>**†.699** | **‡.607**<br>**‡.607** | .615<br>.629 | .499<br>. 534 |
| AIDA | .076<br>.410 | .416<br>.384 | .071<br>.072 | .210<br>.184 | .166<br>.173 | **‡.623**<br>**‡.563** | .331<br>.556 | .069<br>.077 | .353<br>.294 | .404<br>.347 | .617<br>.607 | .303<br>. 333 |
| Babelfy | .517<br>.685 | .543<br>.496 | .668<br>.667 | .520<br>.512 | .364<br>.348 | **†.731**<br>**†.696** | .471<br>.621 | .600<br>.538 | .439<br>.378 | .441<br>.379 | .684<br>.663 | .543<br>. 544 |
| DBSpotlight | .471<br>.664 | .426<br>.436 | .520<br>.502 | .701<br>.675 | .296<br>.279 | .439<br>.401 | .495<br>.660 | .351<br>.333 | .325<br>.255 | .200<br>.161 | .244<br>.200 | .406<br>.415 |
| Dexter | .507<br>.667 | .407<br>.387 | .513<br>.502 | .284<br>.251 | .204<br>.204 | .183<br>.123 | .404<br>.587 | .293<br>.298 | .354<br>.302 | .369<br>.293 | .580<br>.510 | .373<br>. 375 |
| EC-NER | .488<br>.656 | .439<br>.420 | .403<br>.369 | .244<br>.194 | .137<br>.150 | .290<br>.252 | .412<br>.594 | .429<br>.407 | .365<br>.335 | .331<br>.320 | .192<br>.160 | .339<br>.351 |
| Kea | .634<br>.755 | .539<br>.524 | **†.763**<br>**†.753** | **‡.733**<br>**‡.725** | **†.472**<br>.453 | .588<br>.527 | **†.631**<br>**†.758** | **†.662**<br>**‡.615** | .501<br>.447 | .435<br>.387 | **‡.761**<br>**‡.753** | **†.611**<br>**‡.609** |
| NERD-ML | .558<br>.714 | .465<br>.427 | .575<br>.554 | .548<br>.528 | .422<br>.411 | .312<br>.252 | .478<br>.629 | .513<br>.502 | .402<br>.340 | .367<br>.297 | .740<br>.719 | .489<br>.488 |
| TAGME 2 | **†.660**<br>**‡.776** | .513<br>.481 | **‡.723**<br>.708 | .661<br>.642 | .385<br>.372 | .590<br>.532 | .578<br>.712 | .590<br>.556 | .445<br>.380 | .470<br>.391 | **†.832**<br>**†.814** | .586<br>.579 |
| WAT | **‡.643**<br>.758 | **†.597**<br>**†.581** | .714<br>**‡.714** | .653<br>.666 | .401<br>.385 | .593<br>.491 | **‡.601**<br>**‡.740** | .601<br>.542 | .504<br>.427 | .433<br>.364 | .697<br>.648 | .585<br>.574 |
| Macro-Average | .528<br>.694 | .490<br>.473 | .558<br>.547 | .511<br>.497 | .343<br>.338 | .461<br>.409 | .477<br>.653 | .482<br>.462 | .444<br>.404 | .430<br>.384 | .609<br>.586 | |

with comparable query time is TAGME. Nonetheless in this section, we show the accuracy comparison results of PML with 10 other disambiguation systems (including the ones with significantly slower query time) for the sake of completeness.

**Comparison using Wikipedia as Ground Truth:** In this section, we compare the proposed disambiguation system with DBpedia Spotlight[6] and TAGME[7].

An annotation set $\mathcal{G} \subset \mathcal{A}_2$ is used as an input of two Spotlight instances of different confidence values $\gamma = 0.0$ and $\gamma = 0.5$. We note that as Spotlight may not return disambiguation results for intended target mentions in annotations input due to pruning, Spotlight outputs are only for a subset $\mathcal{G}' \subset \mathcal{G}$. We then use the proposed PML disambiguation system of setting $(n_1 = 100, n_2 = 1)$ without pruning. For fairness, we only compare precision results on the subset $\mathcal{G}'$. The results are shown in Table 4, indicating that our proposed system has a higher accuracy of between 2.2% and 8.2% depending on the metric. The precision drop from $\mathcal{P}_{DS}$ to $\underline{\mathcal{P}}_{DS}$ implies that Spotlight disambiguation does not work as well as PML across distinct mentions.

For TAGME, a similar methodology is employed, but with a minor difference: the TAGME web API does not allow the user to specify the annotation for disambiguation. As a result, we rely on the TAGME spotter, and only include results where TAGME annotated exactly the same mention as the ground truth data. The precision results are shown in Table 5, indicating that our proposed system has a higher accuracy from 3.3% to 7.1%.

**Comparison using GERBIL:** To provide convincing evidence that our system works well on more than just Wikipedia text, we also compared our system to 10 other disambiguation systems over 11 different datasets. This was done by implementing a web-based API for our system that is compatible with GERBIL 1.2.2. [24]. Due to space constraints, we refer the interested reader to the GERBIL website[8] and paper [24] for a complete description of these systems and datasets. The task we considered is the *strong annotation task* (D2KB). In this task, we are given an input text containing a number of marked phrases, and in the output, marked phrases are associated with entities from the knowledge base. Note that the systems AGDISTIS, Babelfy, KEA, Spotlight, and WAT support D2KB directly, whereas other systems only support a *weak annotation task* (A2KB). However, GERBIL has a built-in methodology to allow these annotators to take part in the experiment[9].

We tested our system using all datasets available by default in GERBIL, which are primarily based on news articles, RSS feeds, and tweets. In Table 6, we report, for each combination of system and dataset, the micro-F1 (top) and

[6] We used Spotlight 0.7 [4] (statistical model `en_2+2` with the SpotXmlParser.

[7] We used the TAGME version 1.8 web API http://tagme.di.unipi.it/tag in January, 2016.

[8] http://aksw.org/Projects/GERBIL.html.

[9] See the main Gerbil website as well as https://github.com/AKSW/gerbil/wiki/D2KB#handling-of-higher-order-annotators for more details. To quote the GERBIL documentation, "The response of these annotators is filtered using a strong annotation match filter. Thus, all entities that do not exactly match one of the marked entities in the gold standard are removed from the response of the annotator before it is evaluated.".

macro-F1 (bottom) scores. The micro-F1 score is the F1-measure aggregated across annotations, while the macro-F1 score is aggregated across documents. Even though not being trained on such datasets, our system is very competitive to the others.

Firstly, we observe that our system achieves very high macro-F1 scores. These macro-F1 scores are the highest in terms of average (c.f. Fig. 1), .644, and lowest in terms of the average of the ranking among 11 systems (c.f. Fig. 2), 2.45; Kea comes in second with .609 and 2.64 respectively. In terms of micro-F1, we fall slightly short of Kea in terms of average and ranking-average, .611 vs. .600 and 2.72 vs. 3.36, respectively.

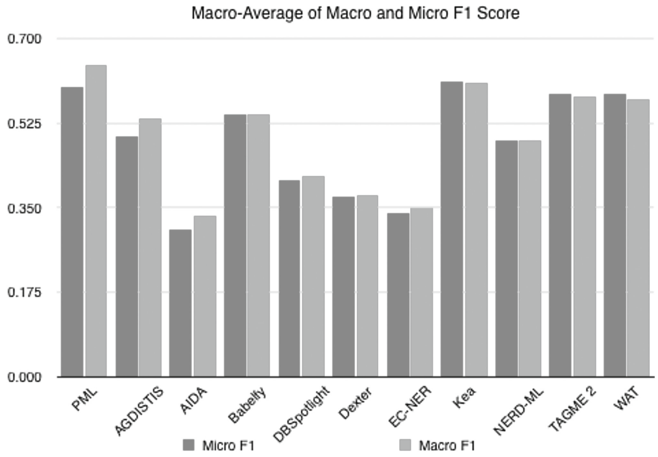

**Fig. 1.** The average of Micro and Macro F1 for different techniques across different data sets in Table 6

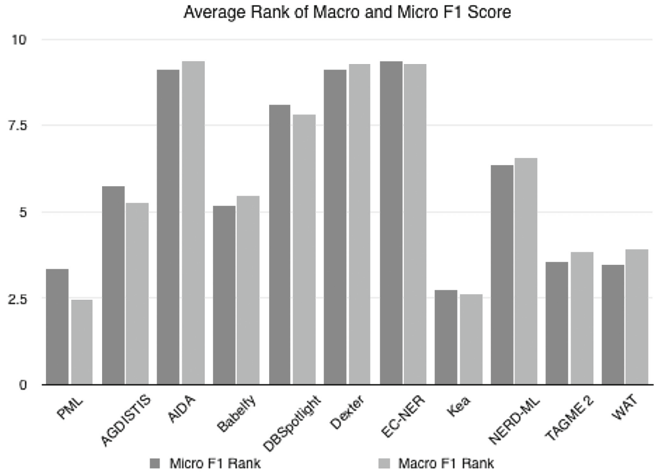

**Fig. 2.** Average rank of Micro and Macro F1 for different techniques (across different data sets in Table 6)

Secondly, our system does very well on news. If we restrict ourselves to the news datasets (ACE2004, AIDA/CoNLL, AQUAINT, MSNBC, N3-Reuters-128, N3-RSS-500), then we achieve the highest average and lowest rank-average scores in terms of both micro-F1 and macro-F1: .661/1.83 and .612/3.

However, our system performs quite poorly on the KORE50 dataset, which is significantly different from the training environment of Wikipedia dataset. Many entries in KORE50 dataset are single sentences involving very ambiguous entities: since our system does not perform joint disambiguation, these highly ambiguous entities are problematic, resulting in a performance drop[10].

## 8 Conclusions

This paper proposes a new per-mention learning (PML) disambiguation system, in which the feature engineering and model training is done per unique mention. The most significant advantage of this approach lies in the specialized learning that is highly parallelizable, supports fast queries and efficient updates. Furthermore, this per-mention disambiguation approach can be easily calibrated or tuned for specific mentions with new datasets, without affecting the results of other mentions.

In a pairwise direct comparison over 30–60 thousands of samples, our system clearly outperforms Dbpedia Spotlight and TAGME. Moreover, under the public benchmark system GERBIL, we have shown that our PML system is very competitive with 10 state-of-the-art disambiguation systems over 11 different datasets, and, for the case of disambiguating news, consistently outperforms other systems. In terms of macro-F1, PML achieves the highest average-score and the lowest average-ranking across all datasets.

## References

1. Brando, C., Frontini, F., Ganascia, J.: REDEN: named entity linking in digital literary editions using linked data sets. CSIMQ **7**, 60–80 (2016)
2. Cucerzan, S.: Large-scale named entity disambiguation based on wikipedia data. In: Proceedings of the EMNLP-CoNLL, pp. 708–716, June 2007
3. Cucerzan, S.: Name entities made obvious: the participation in the ERD 2014 evaluation. In: Proceedings of the ERD, pp. 95–100. ACM, New York (2014)
4. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the I-SEMANTICS (2013)
5. Ferragina, P., Scaiella, U.: TAGME: on-the-fly annotation of short text fragments (by Wikipedia entities). In: Proceedings of the CIKM, pp. 1625–1628 (2010)

[10] Ideally, to achieve better performance, one would need to adapt and retrain supervised models for scenarios with short and dynamic contexts such as KORE50 dataset. One potential issue of such retraining is the lack of big labelled data. This issue could be solved by integrating the target labelled dataset with Wikipedia dataset and adjusting the sample weights to balance the training cost of the target and Wikipedia datasets. However, we decided not to do so to maintain the fairness of this comparison.

6. Ferrucci, D.A.: Introduction to “This is Watson”. IBM J. Res. Dev. **56**(3), 235–249 (2012)
7. Ganea, O., Ganea, M., Lucchi, A., Eickhoff, C., Hofmann, T.: Probabilistic bag-of-hyperlinks model for entity linking. In: Proceedings of the WWW, pp. 927–938 (2016)
8. Guo, Z., Barbosa, D.: Robust entity linking via random walks. In: Proceedings of the CIKM, pp. 499–508 (2014)
9. Han, X., Sun, L.: A generative entity-mention model for linking entities with knowledge base. In: Proceedings of the HLT, pp. 945–954 (2011)
10. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: a graph-based method. In: Proceedings of the SIGIR, pp. 765–774 (2011)
11. Hoffart, J.: Discovering and disambiguating named entities in text. In: Proceedings of the SIGMOD/PODS Ph.D. Symposium, pp. 43–48 (2013)
12. Houlsby, N., Ciaramita, M.: A scalable Gibbs sampler for probabilistic entity linking. In: Rijke, M., Kenter, T., Vries, A.P., Zhai, C.X., Jong, F., Radinsky, K., Hofmann, K. (eds.) ECIR 2014. LNCS, vol. 8416, pp. 335–346. Springer, Cham (2014). doi:10.1007/978-3-319-06028-6_28
13. Hulpuş, I., Prangnawarat, N., Hayes, C.: Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In: Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d’Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 442–457. Springer, Cham (2015). doi:10.1007/978-3-319-25007-6_26
14. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of Wikipedia entities in web text. In: Proceedings of the KDD, pp. 457–466 (2009)
15. McNamee, P.: HLTCOE efforts in entity linking at TAC KBP 2010. In: Proceedings of the TAC (2010)
16. Meij, E., Weerkamp, W., de Rijke, M.: Adding semantics to microblog posts. In: Proceedings of the WSDM, pp. 563–572 (2012)
17. Milne, D., Witten, I.H.: Learning to link with Wikipedia. In: Proceedings of the CIKM, pp. 509–518 (2008)
18. Moro, A., Raganato, A., Navigli, R.: Entity linking meets word sense disambiguation: a unified approach. TACL **2**, 231–244 (2014)
19. Olieman, A., Azarbonyad, H., Dehghani, M., Kamps, J., Marx, M.: Entity linking by focusing DBpedia candidate entities. In: Proceedings of the ERD, pp. 13–24 (2014)
20. Piccinno, F., Ferragina, P.: From TAGME to WAT: a new entity annotator. In: Proceedings of the ERD, pp. 55–62 (2014)
21. Qureshi, M.A., O’Riordan, C., Pasi, G.: Exploiting wikipedia for entity name disambiguation in tweets. In: Proceedings of the NLDB, pp. 184–195 (2014)
22. Suchanek, F., Weikum, G.: Knowledge harvesting in the big-data era. In: Proceedings of the SIGMOD, pp. 933–938. ACM, New York
23. Usbeck, R., Ngomo, A.N., Röder, M., Gerber, D., Coelho, S.A., Auer, S., Both, A.: AGDISTIS - agnostic disambiguation of named entities using linked open data. In: Proceedings of the ECAI, pp. 1113–1114 (2014)

24. Usbeck, R., Röder, M., Ngonga Ngomo, A.-C., Baron, C., Both, A., Brümmer, M., Ceccarelli, D., Cornolti, M., Cherix, D., Eickmann, B., Ferragina, P., Lemke, C., Moro, A., Navigli, R., Piccinno, F., Rizzo, G., Sack, H., Speck, R., Troncy, R., Waitelonis, J., Wesemann, L.: GERBIL: general entity annotator benchmarking framework. In: Proceedings of the WWW, pp. 1133–1143 (2015)
25. Zwicklbauer, S., Seifert, C., Granitzer, M.: Robust and collective entity disambiguation through semantic embeddings. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 425–434. ACM (2016)