Learning text representations for 500K classification tasks on Named Entity Disambiguation

Ander Barrena and **Aitor Soroa** and **Eneko Agirre**
IXA NLP Group
UPV/EHU University of the Basque Country
Donostia, Basque Country
ander.barrena,a.soroa,e.agirre@ehu.eus

Abstract

Named Entity Disambiguation algorithms typically learn a single model for all target entities. In this paper we present a word expert model and train separate deep learning models for each target entity string, yielding 500K classification tasks. This gives us the opportunity to benchmark popular text representation alternatives on this massive dataset. In order to face scarce training data we propose a simple data-augmentation technique and transfer-learning. We show that bag-of-word-embeddings are better than LSTMs for tasks with scarce training data, while the situation is reversed when having larger amounts. Transferring an LSTM which is learned on all datasets is the most effective context representation option for the word experts in all frequency bands. The experiments show that our system trained on out-of-domain Wikipedia data surpasses comparable NED systems which have been trained on in-domain training data.

1 Introduction

Named Entity Disambiguation (NED), also known as Entity Linking or Entity Resolution, is a task where entity mentions in running text need to be linked to its entity entry in a Knowledge Base (KB), such as Wikidata, Wikipedia or other derived resources like DBpedia (Bunescu and Pasca, 2006; McNamee and Dang, 2009; Hoffart et al., 2011). This task is challenging, as some entity mentions like "London" can refer to a number of places, people, fictional characters, brands, movies, books or songs.

Given a mention in context, NED methods (Cucerzan, 2007; Han and Sun, 2011; Ratinov et al., 2011; Lazic et al., 2015) typically rely on three models: (1) a mention model which collects possible entities which can be referred to by the mention string (aliases or surface forms), possibly weighted according to prior probabilities; (2) a context model which measures to which extent the entities fit well in the context of the mention, using textual features; (3) a coherence model which prefers entities that are related to the other entities in the document. The first and second models are *local* in that they only require a short context of occurrence and disambiguate each mention in the document separately. The third model is *global*, in that all mentions are disambiguated simultaneously (Ratinov et al., 2011). Recent work has shown that local models can be improved adding a global coherence model (Ratinov et al., 2011; Globerson et al., 2016). In this work we focus on a local model, and a global model could improve the results further.

All local and global systems mentioned above, as well as the current state-of-the-art systems (Lazic et al., 2015; Globerson et al., 2016; Yamada et al., 2016; Ganea and Hofmann, 2017), rely on single models for each of the above, that is, they have a single mention model, context model and coherence model for all entities, e.g. the 500K ambiguous entity mentions occurring more than 10 times in Wikipedia. While this has the advantage of reusing the parameters across mentions, it also makes the problem unnecessarily complex.

In this paper we propose to break the task of NED into 500K classification tasks, one for each target mention, as opposed to building a single model for all 500K mentions. The advantage of this approach is that each of the 500K classification tasks is simpler, as the classifier needs to focus on learning a good context model for a single mention and a limited set of entities (those returned by the mention model). On the negative side, training instances for mentions follow a long tail distribution, with some mentions having a huge number of examples, but with the vast majority

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 171–180
Brussels, Belgium, October 31 - November 1, 2018.

of mentions having very limited training data, e.g. 10 occurrences linking to an entity in Wikipedia. Our results will show that data-augmentation and transfer learning allow us to overcome the sparseness problem, yielding the best results among local systems, very close to the best local/global combined systems. Contrary to systems trained on in-domain data (Cucerzan, 2012; Chisholm and Hachey, 2015; Globerson et al., 2016; Yamada et al., 2016; Sil et al., 2018), ours is trained on Wikipedia and tested out-of-domain.

From another perspective, a set of 500K classification problems provides a great experimental framework for testing text representation and classification algorithms. More specifically, deep learning methods provide end-to-end algorithms to learn both representations and classifiers jointly (LeCun et al., 2015). In fact, learning text representations models has become a center topic in natural language understanding, as it allows to transfer representation models across tasks (Conneau and Kiela, 2018; Peters et al., 2018; Wang et al., 2018). In this paper, we explore several popular text representation options, as well as data-augmentation (Zhang and LeCun, 2015) and transfer learning (Bengio, 2012). All training examples and models in this paper, as well as the pytorch code to reproduce results is availabe [1].

This paper is structured as follows. We first present our models. Section 3 presents the experiments, followed by related work and conclusions.

2 Deep Learning models for NED

In this section we describe the deep learning models proposed in this work. We first present our use of Wikipedia to produce the candidate model and the training instances, followed by the deep learning models. We will mention options and hyper-parameters as we explain each component. Unless explicitly stated we used default values. The rest were selected and tuned solely on development data from Wikipedia itself, with no access to other datasets (cf. Section 3).

2.1 Pre-processing and Resources

We used the English Wikipedia[2] as the only resource for training the models. On the one hand, Wikipedia articles define the target set of entities. On the other hand, Wikipedia editors have manually added hyperlinks to articles, where the anchor text corresponds to the mention, and the url corresponds to the entity.

We first built a candidate model as a dictionary that links each text anchor to possible entities, using the method presented in (Spitkovsky and Chang, 2012; Barrena et al., 2016). Let M be the set of all unique mention strings m, E the set of all target entities e, and $E_m = \{e_1, \ldots, e_m\}$ the set of entities that can be referred by mention m. We kept the 30 most frequent candidates for each mention for the sake of efficiency. We report the sizes of E and M below.

We then extracted annotated examples by scanning through the page contents for hyperlinks that link anchors (the mentions) to the corresponding Wikipedia pages (the entities). For each such hyperlink, we build a context $\mathbf{c}$ by first tokenizing and removing the stop words, and then extracting a window of 20 words to the left and 20 words to the right from the anchor. We thus construct a set of N_m labeled instances $\{\mathbf{c}_i, y_i\}$, where $y_i \in E_m$, for each m. We did not apply any kind of lemmatization or stemming to the training contexts.

2.2 Word Expert models

In the word expert approach we train one classifier for each possible ambiguous mention. We are thus interested in learning a classifier that assigns a target mention $m \in M$ appearing in a context $\mathbf{c}$ to one of its possible entity candidates $E_m = \{e_1, \ldots, e_m\}$ based on the set of N_m training instances $\{\mathbf{c}_i, y_i\}$, where $y_i \in E_m$. From the approximately $1M$ ambiguous mentions in Wikipedia only $523K$ occur more than 10 times as anchors in Wikipedia, and we thus limit M to 523 mentions and learn $523K$ classifiers.

Given the textual context of a mention, $\mathbf{c}_i$, the text representations model will output a vector representation h. We tried different alternatives for representing context, as described below. Given the vector h, we define the classifier as a neural network consisting of a number of fully connected layers, followed by a softmax layer with as many output dimension as the number of candidates of the target mention E_m. The whole network (representation model and classifier) is trained end-to-end using cross-entropy loss.

In order to tune the hyper-parameters, we split the examples into training (90%) and development

[1] https://github.com/anderbarrena/500kNED

[2] We chose the 2014 snapshot, which gives good results in the contemporary evaluation datasets.

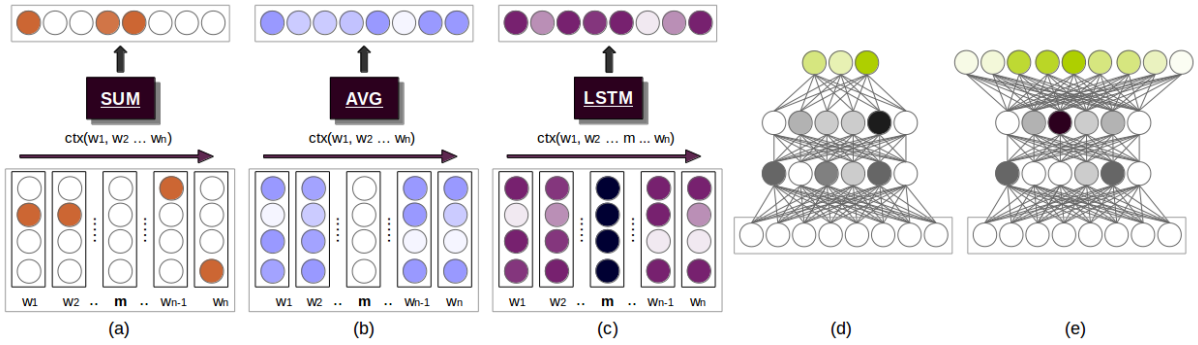

Figure 1: Deep learning models for NED. On the left side, context models: (a) Sparse BoW, (b) Continuous BoW, (c) LSTM. On the right side the classification models: (d) word expert model, (e) single model. The transfer model first learns an LSTM on the single model, then reuses the LSTM to learn each of the word expert models.

(10%). We tried different configurations of the classifier, such as the number of fully connected layers or the activation function. Two layers of ReLUs performed best in the development set. The rest of parameters were set by default: 256 hidden units, adam optimizer with an initial learning rate of $1.0e-3$ and batches of 256 instances. Training stops when the accuracy in the development set drops for 10 consecutive epochs or when a maximum of 300 epochs is reached. We select the model that obtained the best accuracy in the development set before stopping. The same parameters and model were used for all word experts.

We now describe the how to represent context.

Sparse bag-of-words (BoW): In this model, depicted in Figure 1 (a), the context is represented as the addition of the one-hot vector for each word, with as many dimensions as the vocabulary size. The target mention is assigned a zero vector, akin to ignoring it. The vocabulary is large, comprising more than 200K different words, which slowed down learning. Alternatively, we also clustered the words in the vocabulary. In this case, we use those clusters to represent the words in the one-hot vector, yielding a bag of clusters representation. We used the *word2vec*[3] toolkit to build the clusters from English Wikipedia text. The corpus was lower-cased and tokenized. We found that using 3K cluster size does best in development.[4] As the results on development for the models using words were below those of clusters we will report only results for clusters.

Continuous bag-of-words (CBoW): In this case, see Figure 1 (b), context is represented with the centroid of pre-trained word embeddings, where the mention is represented by a vector of zeros. The embeddings were trained over the English Wikipedia using `word2vec` (Mikolov et al., 2013). The corpus was first lower-cased, and we used a window size of 20, 10 negative samples and 7 iterations. The embeddings have a dimensionality of 300. We also tested a number of pre-trained embeddings, but we did not obtain better results, perhaps because our embeddings were trained on Wikipedia, which is also the training corpus for the NED system. When combined with the classifiers, we kept the embeddings fixed.

Recurrent Neural Network (LSTM): As a third alternative, we considered a recurrent neural network based on LSTMs (Hochreiter and Schmidhuber, 1997) to exploit the dependencies among the word sequence that forms the input context (Figure 1, (c)). We use a single LSTM to encode the input contexts as follows. We first replace the target mention with a special symbol which has a manually assigned constant embedding, and then feed the sequence into the LSTM. The last hidden vector is taken to represent the context. The LSTMs have 512 hidden units and 300 dimensional word embeddings, which are initialized with the embeddings vectors used in the continuous BoW model described above. The LSTM layers have a dropout layer, with 0.2 dropout probability.

We explored GRUs, stacking LSTMs, temporal average and max pooling among hidden states, but did not improve results on development.

2.3 Single model

One of the main problems of word experts is that they need a large number of manually annotated

[3] https://code.google.com/archive/p/word2vec/

[4] We tried 100,300,800,1K,3K,8K and 10K cluster sizes.

examples for each possible mention, which makes it unsuitable for less frequent mentions. As an alternative, we also trained a single model. Given the set of all training instances N_m for all possible mentions $m \in M$, we train a classifier that, for each context $\mathbf{c}_i$ produces the correct entity $e_i \in E$. This classifier has a large number of classes $|E|$. We discarded entities with less than 50 mentions, and gather up to 5K random instances for the rest. Note that clipping the instance number to 5K effectively downsamples those entities that are highly frequent. All in all, we gather a training corpus of 53M annotated examples for 248K target entities in this single model.

We adopted the recurrent model presented in the previous section. In this case, we also replace the target mention with a special symbol which has a manually assigned constant embedding vector, we feed it into the LSTM, and use the last hidden vector h as the context representation. The classifier follows the same architecture as the word expert model. In this case the LSTM has 2048 hidden units, producing a 512-dimensional context representation and 300 dimensional word embeddings, which are again initialized with the pretrained embeddings in the previous section keeping them fixed. The final softmax layer has 248K dimensions, the number of candidate entities. We checked other hidden-unit sizes with no better results. In order to improve results, we filter out the candidates which are not in dictionary.

Regarding the training details, we use the Adam optimization algorithm with an initial learning rate of $1.0e^{-4}$, and a dropout value of 0.2. In this case, we used a 1% sample of Wikipedia instances as a validation set, and we stop early, whenever the accuracy in this validation set does not improve for 3 consecutive epochs. Training the model takes around 16 hours per epoch in a single GPU, taking at most 18 iterations.

2.4 Transfer learning

As an alternative to learning a single model, we can use the text representation layer of the aforementioned single model in the word expert model. That is, after training the single model with the whole Wikipedia, we use the learned model of the LSTM as the text representation layer of the word expert models. This way, we reuse the LSTM which was learned alongside the single model instead of learning a separate LSTM layer for each word expert (see Section 2.2). When training the word experts, we keep the LSTM layer fixed.

2.5 Data augmentation

As mentioned above, some mentions only have as few as 10 training instances. In order to have a larger number of training instances, we augment the training set for target mention m with the contexts of other mentions that occur as anchors of one of the e_i candidates m ($e_i \in E_m$). Using this strategy, we randomly select up to $250K$ examples as training instances for each mention. Although augmenting the training set has the advantage of providing more training instances, it also has the drawback of distorting the number of examples for each entity. For instance, in the case of the mention *EU* most of the examples in Wikipedia refer to the European Union (around 2800) and only a few to Europe (the continent, 716). When augmenting the training set with examples for the entities, we add more examples for Europe (around 44000) than for European Union (around 13000), changing the ratio of labels in the training data significantly. In order to counter-balance this effect, we tried to combine the priors from the original data with the output of the classifier trained with the augmented dataset. Alternatively, we combined both original and augmented classifiers, yielding better results in development. We thus train two classifiers for each mention, one using the original training set $P(e|\mathbf{c})_{orig}$, and one using the augmented dataset $P(e|\mathbf{c})_{aug}$. Finaly, we combine their scores to produce the combined output $P(e|\mathbf{c})$:

$$P(e|\mathbf{c}) = P(e|\mathbf{c})_{orig} P(e|\mathbf{c})_{aug} \quad (1)$$

3 Experiments

We developed and evaluated our model in standard datasets for easier comparison to the state of the art. The main dataset is the Aida CoNLL dataset which is composed of news documents from the Reuters corpus. It comprises three parts: *Aida-train* training set, *Aidatesta* development set and *Aidatestb* test set. We also include the three earliest Text Analysis Conference (TAC) datasest, that focus on highly ambiguous mentions from news, web and discussion forums: *Tac2010*, *Tac2011* and *Tac2012*. As we are interested in building a robust model based on Wikipedia alone, we ignore the training data accompanying each dataset. We

	testa	testb	tac2010	tac2011	tac2012
mentions	5917	5616	2250	2250	2229
inKB mentions	4792	4485	1020	1124	1177
uniq mentions	2600	2441	750	1315	781
uniq inKB mentions	1850	1685	386	628	509
inKB mentions in dict	1841	1675	382	597	499

Table 1: Statistics of the datasets (see text for details).

use *Aidatesta* for model selection only (i.e. the parameters were tuned on a subset of Wikipedia, cf. Section 2), and *Aidatestb*, *Tac2010*, *Tac2011* and *Tac2012* for out-of-domain test.

Note that we used *Aidatesta* only to select the best models, given that all hyperparameters where tuned over Wikipedia itself. Table 1 shows the statistics for all datasets. From all mentions, only a subset of them actually refers to an entity in the KB provided by the dataset authors ("inKB mentions" row). Our dictionary covers most but not all of those KB entities ("uniq inKB mentions in dict" row). Some of the mentions in the datasets are resolved as NIL, for cases where the mention refers to an entity which is not in the KB. The simplest method to return NIL is to first resolve over all Wikipedia entities, and if the selected entity is not in the KB then to return NIL. We focus the evaluation in the mentions linked to an entity in the respective KB, and use the so-called *inKB accuracy* as the evaluation measure, which is defined as the fraction of correctly disambiguated mentions divided by the total number of mentions which are linked to the KB. We perform 3 runs for each reported result, reporting mean accuracy and standard deviation values. We also include MFS baselines in the results: given a mention, the baseline is computed assigning the entity in the dictionary with highest prior probabilities.

During testing, given a mention, we search the document and try to find the longest string that a) contains the mention and b) matches an entry in the dictionary. Next, we replace every mention string with that longer string in the document.[5] We also apply the *'One entity per Document'* hypothesis, averaging the results of the occurrences for the same mention in the same document (Barrena et al., 2014).

3.1 Development results

Table 2 shows the performance of each of the context representation models and data augmentation options in *Aidatesta*. The MFS baseline obtains 71.91, which is a good point of comparison to benchmark our candidate model (the dictionary) with respect to other systems. All our models improve over the MFS baseline by a large margin. As mentioned in Section 2.5, we have three classifiers for each mention. $P(e|\mathbf{c})_{orig}$ uses the original training set, $P(e|\mathbf{c})_{aug}$ uses the augmented training set, and $P(e|\mathbf{c})$ combines both. The table shows that the results of the original and augmented classifiers are more or less comparable, while the combination consistently yields the best results for all context representations options.

Regarding the representation models, we can observe that the sparse Bag-of-word model yields worse results than the continuous Bag-of-words. The LSTMs learned separately do not improve over continuous BoW, while the LSTM transferred from the single model obtains the best results. In addition to the results in the table, the single model obtains an accuracy of 45.95, well below the rest.

These results confirm our intuitions. Regarding the single model vs. word experts, the classifier has a much easier task in the second case, as the number of classes to predict is much smaller for each classifier. Regarding the performance of the word expert LSTMs, our hypothesis was that, given the long tail distribution of the number of training instances, the per-mention LSTMs of many mentions would not have enough training instances to learn effective representations. We checked this hypothesis plotting the results for each method according to the number of training instances. Figure 2 shows the inKB accuracy for mentions bucketed according to the number of training instances[6]. Continuous BoW overpeforms LSTMs on mentions with a small number of

[5] Mentions that are named as a DBPedia entity classified as location are not expanded.

[6] We set 10 buckets with an equal number of mentions in each bucket

	Sparse BoW	CBoW	LSTM	Transfer
$P(e\|c)_{orig}$	79.65±0.06	82.48±0.48	80.35±0.05	84.70±0.06
$P(e\|c)_{aug}$	79.54±0.26	81.74±0.21	80.66±0.26	82.39±0.42
$P(e\|c)$	83.28±0.17	86.19±0.19	84.35±0.30	**86.87**±0.14

Table 2: Development results (Aidatesta) as inKB accuracy and standard deviation for Sparse BoW, Continuous BoW, LSTM and transferred LSTMs. Each row corresponds to the original training data, augmented training data, and combination.

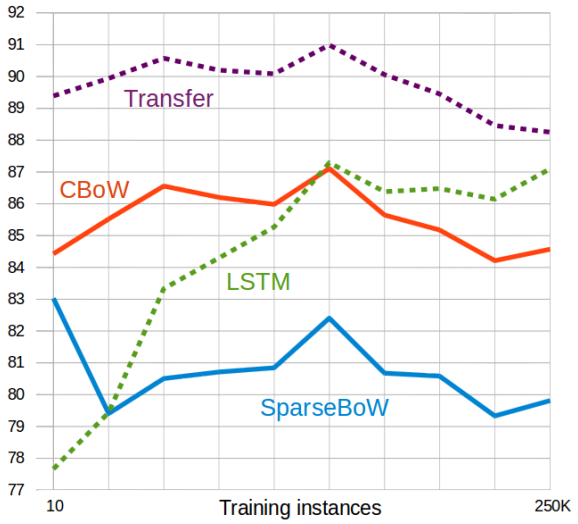

Figure 2: Development results (Aidatesta) as inKB accuracy according to number of training instances.

Method	testb
Local models	
(Lazic et al., 2015) sup.	79.7
Sparse BoW	86.72±0.23
Continuous BoW	89.39±0.44
LSTM	88.44±0.26
Transfer LSTM	**91.19**±0.07
(Lazic et al., 2015)† semi-sup.	86.4 †
(Yamada et al., 2016)*	87.2*
(Ganea and Hofmann, 2017)*	88.8*
Local & Global models	
(Chisholm and Hachey, 2015)*	88.7*
(Globerson et al., 2016)*	91.0*
(Yamada et al., 2016)*	91.5*
(Ganea and Hofmann, 2017)*	92.2*

Table 3: Test results on *Aidatestb* as inKB accuracy. * for systems trained on in-domain data. † for systems using semi-supervised methods.

training instances, while the situation is reversed for mentions with a large number of training instances. The graph also shows that the transferred LSTM yields better results for all frequencies, and that the Sparse BoW model underperforms the rest of models consistently. As an aside, we observed that for the words which have more than 200.000 training instances, both the per-mention LSTM and the transferred LSTM yield similar results.

3.2 Final results

In this section we compare our system with the state-of-the-art in Named Entity Disambiguation. Given the vast number of NED systems, we only report the results of the most relevant high performing systems only. Note that, contrary to us, many high-performing systems use in-domain training data (Ganea and Hofmann, 2017; Globerson et al., 2016), and/or external candidates and link counts when building the dictionary (Lazic et al., 2015; Globerson et al., 2016; Yamada et al., 2016).

Table 3 shows the inKB results on *Aidatestb*, the most popular evaluation dataset. The results show that all our models improve over locals out-of-domain systems trained solely on Wikipedia, but, most notably, also over in-domain systems which were trained on *Aidatrain* (marked with *) and the semi-supervised system (marked with †), which uses large numbers of un-annotated data. As expected, the relative performance of our systems is the same as in development.

All the systems included in Table 3 (except Lazic et al., 2015) use the "means" tables of YAGO as candidates, as this was the entity inventory used by the developers of the dataset (Hoffart et al., 2011). In our case, as we link mention to Wikipedia entities, we just ignore those entities not belonging to the YAGO "means" table. In order to provide head-to-head comparison, the results of our best system when not using the YAGO information is 89.93, more than three points better.

Table 4 shows the inKB accuracy results on the three TAC datasets. In this case, the dataset is accompanied by a KB which is a subset of the Wikipedia 2008 snapshot. Following standard procedure (Globerson et al., 2016), we fil-

Method	tac10	tac11	tac12
Local models			
(Lazic et al., 2015) sup.	—	74.5	68.7
Sparse BoW	85.82	80.25	63.12
Continuous BoW	86.96	81.55	67.49
LSTM	86.73	81.44	67.32
Transfer LSTM	**87.32**	**84.41**	**72.58**
(Lazic et al., 2015)† semi-sup.	—	79.3†	74.2†
(Chang et al., 2016)*	84.5*	—	—
(Yamada et al., 2016)*	84.6*	—	—
Local & Global models			
(Cucerzan, 2012)*	—	—	72.0*
(Chisholm and Hachey, 2015)*	80.7*	—	—
(Globerson et al., 2016)*	87.2*	84.3*	82.4*
(Yamada et al., 2016)*	85.2*		

Table 4: Test results on TAC datasets as inKB accuracy. * for systems trained on in-domain data. † for systems using semi-supervised methods.

ter out entities not listed in the KB before evaluating the results. The table shows that the relative performance of our systems is stable. Our best system outperforms the results of the local system trained on Wikipedia on both Tac2011 and Tac2012 (10 and 3 points). Regarding the comparison with other local systems (in-domain and semi-supervised), our results are the best, including global methods. The only exception is on the TAC2012 dataset, where the mentions were short and known to be specially challenging. In fact, the winner of the task (Cucerzan, 2012) performed an especial effort on finding longer correferent mentions in the document. In the case of (Lazic et al., 2015; Globerson et al., 2016), they use a coreference resolver, which could explain their better results on this dataset.

Note that in this section we do not report results of systems which use the candidate dictionary of (Pershina et al., 2015). As observed by (Globerson et al., 2016), among others, that candidate dictionary has been manually pruned and extended to contain the gold standard entity, yielding a dictionary that has a 100% upperbound and very limited ambiguity. This makes the results of systems using this dictionary look much better than those using automatically constructed candidate models. We thus miss results from some papers (Pershina et al., 2015; Sil et al., 2018), and report the results using automatically constructed dictionaries for the rest (e.g. Globerson et al., 2016).

4 Related Work

In this section we will briefly review NED systems and text representation literature. Hachey et al. (2012) present a detailed overview of all possible components, but in this section we will focus on the most relevant high performing systems. Please see (Ling et al., 2015) for a more detailed review of past research.

4.1 NED systems

Among local systems that are trained on Wikipedia alone, (Lazic et al., 2015) was the best performing one to date. Their system is based on probabilistic estimation, with a rich preprocessing pipeline, including dependency parsing, common noun phrase identificacion and coreference resolution. They present the results for both a supervised version, and a graph-based semi-supervised extension which improves results. We think that the results of our method could be improved using richer pre-processing, specially the use of coreference to find longer coreferent mentions, which reduces the ambiguity of the mention and improve results.

Among global models, (Chisholm and Hachey, 2015) use a learning to rank algorithm which combines local and global features, trained on in-domain corpora (*Aidatrain* and *Tac2010 train*, respectively). They improve the results significantly by extending the information extracted from Wikipedia with a web crawl.

In (Yamada et al., 2016), the authors jointly learn word and entity embeddings using Wikipedia. The similarity of word and entity embeddings are used as features to train a Gradient Boosted Regression Trees on in-domain data. They report both local and global results, with a clear improvement when adding a global component.

Ganea and Hofmann (2017) also present a local and global algorithm. In their local algorithm, they combine word and entity embeddings with an attention mechanism trained on in-domain data. The global component is Loopy Belief Propagation, which optimizes the global sequence coherence initialized by the local algorithm. They report the best results among both local and global algorithms in *Aidatestb*, but, unfortunately they don not provide results on the TAC datasets. Given that their global algorithm yields an improvement of 3 points, and that our local method exploits complementary information, we would like to combine both in future work.

Globerson et al. (2016) add a global compo-

nent to Plato (Lazic et al., 2015), whose weights are used to initialize a multi-focal attention mechanism. The global model is trained and optimized on in-domain training datasets. They report the best performance for *TAC* datasets to date, and very good results on *Aida*. Their very strong results on TAC 2012 (together with those of Lazic et al., 2015) seem to be due to the use coreference in the candidate model, as this dataset includes shorter target mentions than the rest.

More recently, Sil et al. (2018) introduce a deep neural cross-lingual entity linking system using a combination of CNN, LSTM and NTNs, with strong results. Their method performs similar to ours on TAC2010, but using the manually curated dictionary of (Pershina et al., 2015), which, as stated before, greatly simplifies the task (c.f. Section 3.2).

All NED systems mentioned above build a single model for all possible target mentions. The only word expert approach that we are aware of is briefly mentioned in (Chang et al., 2016). This paper compares NED and word sense disambiguation, and builds a bag of words logistic regression classifier for each mention. Their results on the *TAC2010* dataset is 84.5, below our results.

4.2 Text representation

Text representation for deep learning is a hot topic on natural language processing (LeCun et al., 2015), and several evaluation frameworks have been proposed (Conneau and Kiela, 2018; Wang et al., 2018). Our 500K classification tasks can be seen as an additional large-scale testbed for text representation proposals.

In a setting similar to ours, (Yuan et al., 2016; Peters et al., 2018) propose to train a language model based on LSTMs and then use it for word sense disambiguation. Instead of using the context representations to learn a classifier directly as we do, they use label propagation in representation space. In our case, instead of using a language model, we train the text representation model on a more closely related task, i.e., that of disambiguating all possible entities.

While bags of pre-trained word embeddings and LSTMs are the most popular approaches for text representation, many alternatives exist. For instance, ELMo (Peters et al., 2018) obtains word embeddings that include contextual information, and then combine them using bag-of-words or other alternative. Alternatively, universal sentence encoding models that are useful in many tasks are being proposed (Arora et al., 2017; Logeswaran and Lee, 2018; Subramanian et al., 2018; Cer et al., 2018). We think that, in supervised classification tasks such as ours, the transferred LSTM already captures well contextual information and that the performance bottleneck might lie on the classifier. If that is the case, stronger context representation models might not make much of a difference. We plan to explore this in future work.

5 Conclusions and Future Work

In this paper we propose to break the task of NED into 500K classification tasks, one for each target mention, as opposed to building a single model for all 500K mentions. The advantage of this word expert approach is that each of the 500K classification tasks is simpler. On the negative side, scarcity of training data is made worse. We show that this problem can be effectively alleviated with data-augmentation and specially with transfer learning.

A set of 500K classification problems provides a great experimental framework for testing text representation and classification algorithms. Given the scarce data available, learning a classifier directly on a bag-of-words or LSTM representation yields weak results. Bringing in pre-trained embeddings improves results, but the key to strong performance is to learn a single model for all entities using an LSTM and then transfer the LSTM to each of the word experts.

Our model is a local system using Wikipedia information alone, yielding the best results among local systems, comparable to systems trained on in-domain data and incorporating global coherence models. All training examples and models in this paper, as well as the pytorch code to reproduce results is availabe [7].

For the future, the performance of our system can be easily improved combining it with a global method such as (Ganea and Hofmann, 2017). There are also specific improvements that can be done, such as using correference (Lazic et al., 2015) or additional information from web crawls (Chisholm and Hachey, 2015). Regarding the use of in-domain training, we think that our out-of-domain results reflect the most realistic scenario, as in-domain training data is rare in practice.

[7] https://github.com/anderbarrena/500kNED

Regarding text representation, we tested some straightforward alternatives. Recent work has proposed stronger options which could improve the results of our word experts further.

Acknowledgments

This research was partially supported by the Spanish MINECO (TUNER TIN2015-65308-C5-1-R, MUSTER PCIN-2015-226, cofunded by EU FEDER), the UPV/EHU (excellence research group), and the NVIDIA GPU grant program.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Ander Barrena, Eneko Agirre, Bernardo Cabaleiro, Anselmo Peñas, and Aitor Soroa. 2014. "one entity per discourse" and "one entity per collocation" improve named-entity disambiguation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2260–2269, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ander Barrena, Aitor Soroa, and Eneko Agirre. 2016. Alleviating poor context with background knowledge for named entity disambiguation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1903–1912, Berlin, Germany. Association for Computational Linguistics.

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36.

R. C. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 9–16, Trento, Italy. The Association for Computer Linguistics.

D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil. 2018. Universal Sentence Encoder. *ArXiv e-prints*.

Angel Chang, Valentin I. Spitkovsky, Christopher D. Manning, and Eneko Agirre. 2016. A comparison of named-entity disambiguation and word sense disambiguation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

A. Conneau and D. Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. *ArXiv e-prints*.

S. Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic.

Silviu Cucerzan. 2012. Msr system for entity linking at tac 2012. In *Text Analysis Conference - Knowledge Base Population 2012 TAC-KBP 2012*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, Berlin, Germany. Association for Computational Linguistics.

B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J.R. Curran. 2012. Evaluating Entity Linking with Wikipedia. *Artificial Intelligence*, 194:130–150.

X. Han and L. Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 945–954, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

J. Hoffart, M.A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Paul McNamee and Hoa Dang. 2009. Overview of the TAC 2009 Knowledge Base Population track. In *TAC*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, Denver, Colorado. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

L.A. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1375–1384. The Association for Computer Linguistics.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *AAAI2018*.

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for english wikipedia concepts. In *LREC*, pages 3168–3175.

S. Subramanian, A. Trischler, Y. Bengio, and C. J Pal. 2018. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. *ArXiv e-prints*.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ArXiv e-prints*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, Berlin, Germany. Association for Computational Linguistics.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1374–1385.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.