Named Entity Disambiguation at Scale

Ahmad Aghaebrahimian and Mark Cieliebak

Institute of Applied Information Technology
Zurich University of Applied Sciences ZHAW
Winterthur, Switzerland
{agha,ciel}@zhaw.ch

Abstract. Named Entity Disambiguation (NED) is a crucial task in many Natural Language Processing applications such as entity linking, record linkage, knowledge base construction, or relation extraction, to name a few. The task in NED is to map textual variations of a named entity to its formal name. It has been shown that parameter-less models for NED do not generalize to other domains very well. On the other hand, parametric learning models do not scale well when the number of formal names expands above the order of thousands or more. To tackle this problem, we propose a deep architecture with superior performance on NED and introduce a strategy to scale it to hundreds of thousands of formal names. Our experiments on several datasets for alias detection demonstrate that our system is capable of obtaining superior results with a large margin compared to other state-of-the-art systems.

Keywords: Named Entity Disambiguation · Alias Detection · Deep Learning.

1 Introduction

Named Entity Disambiguation (NED) [14, 27] is the task of linking textual variations of Named Entities (NE)[1] to their target names, which are usually provided as a list of formal names. For instance, while recognizing "Philip Morris" as an NE is the job of a Named Entity Recognition (NER) system, associating it to "Philip Morris International Inc (PMI)" in a list of formal names as a means of disambiguation is performed via NED. The list of formal names often contains many other names such as "Phil Moors, Morris Industries, ..." which are very similar to the correct formal name and should not be mistaken with it. The number of formal names, which may go to several hundred thousands or even millions, makes NED a challenging task. Character shift, abbreviation, word shift, typos, and use of nicknames are other challenges in NED.

There are two broad categories of approaches to address NED, namely classic and modern. The classic or parameter-less approach [17, 21] is simply a textual similarly function such as the Levenshtein distance, Cosine similarity, or longest common subsequent [10] that computes the pairwise similarity score for all available formal names given each source name.

The time complexity of these models are mostly of the order of $O(f(mn))$, m and n being the number of the source and formal names, respectively, and $f(.)$ being a

[1] Named Entities are well-known places, people, organizations, ...

linear function, which makes them quite fast. Moreover, they are highly parallelizable since computing the score of a batch does not affect the next batch scores. However, the performance of these models severely suffers when porting to new domains [6, 3, 28].

The modern or parametric learning models have better performance in working across domains through fine-tuning or transfer learning and are of the same order of complexity except for $f(.)$, which is often a more complex function. In real-life NED systems, the number of formal names exceeds hundreds of thousands or even million names, which makes a parametric pairwise comparison difficult if not infeasible.

To address these issues, we integrate a deep learning model with a parameter-less method of similarity assignment to break the limit on recognizing new domains and simultaneously scaling the system to millions of names. To this end, we train a term frequency-inverse document frequency (tf-idf) model on a range of character n-grams of all names. The tf-idf model has two tasks; to generate feature vectors for the deep learning model and to set a threshold for limiting the formal names when using the system at inference time. We test the system on four datasets for alias detection [27] and compare the results with several baselines as well as a state-of-the-art NED system.

The motivation for this work for us is to solve a business need with a scalable and efficient solution based on deep neural networks. Our business partner harvests around 100k news articles per day. They want to recognize company names in the news and to link each of them to its formal name available in a proprietary knowledge base which contains almost 80k formal names. The variance between formal names and their usage in the news and the number of formal names in addition to the sheer amount of news articles per day ask for an efficient and scalable system for performing the task.

The main contributions of this work are a deep architecture for scoring entity names and a strategy for leveraging this architecture to a large list of source and/or formal names.

2 Related Work

Before the advent of modern and neural learning models, parameter-less computation of string similarity such as the Cosine similarity, the Levenshtein distance, and the scores proposed by [24] were popular means of scoring formal names given source names. Many of these works use a sort of word-level or character-level n-gram features [20], syntactic features [7], or alignment features [25].

The earliest modern models of NED are based on feature engineering on a classifier such as Support Vector Machines (SVM) [4] coupled with a sequence decoder such as Conditional Random Fields (CRF) [9].

Most advanced neural models today use CRFs for making inference but instead of doing feature engineering manually, they use a form of Deep Neural Network (DNN) such as Long Short Term Memory (LSTM), Convolutional Neural Network (CNN), or Gated Recurrent Unit (GRU) for automatic feature learning [2, 18].

Designing a NED system by feature engineering is a highly time-consuming process, hence end-to-end neural systems capable of learning the features on their own [16, 29] are more approachable systems. All of these models use a form of neural similarity function on top of the entity embeddings, mostly on the token-level and in some studies on character-level [12]. However, the inference module in these models is usually a pairwise scoring method [1] against all formal entities given each source entity, which makes the inference unpractical for applications with a large number of formal names.

NED can be performed jointly with NER in a way that the errors generated by NER are recovered by NED. A common approach for jointly training NED with NER is to use a NER system to extract entity mentions and use feature engineering in a shared space to map the source entities to their formal names [23, 19]. The number of formal names is a limiting factor for these systems, too.

When the number of formal names is limited, NED is usually done as a single NER process. State-of-the-art NER systems [11] use a form of pre-trained embeddings that is fed into a form of Recurrent Neural Network (RNN). The resulting representations are used to form a trellis for a Conditional Random Field (CRF) [15] decoder which extracts the beginning and the end tokens of named entities. However, when the number of formal names increases, besides the lack of enough training data, the CRF turns intractable.

Finally, [22] proposed an architecture using a Multi-Layered Perceptron (MLP) to recognize toponyms, and similar neural network architecture is used by [27] for entity linking. Our work is similar to these two last studies, but our pair-wise ranking architecture is coupled with a strategy that allows us to leverage the disambiguation to millions of source and formal names by filtering irrelevant formal names out.

3 Model Description

We model text similarity as a softly constrained pair-wise ranking problem. Figure 1 schematically represents the model.

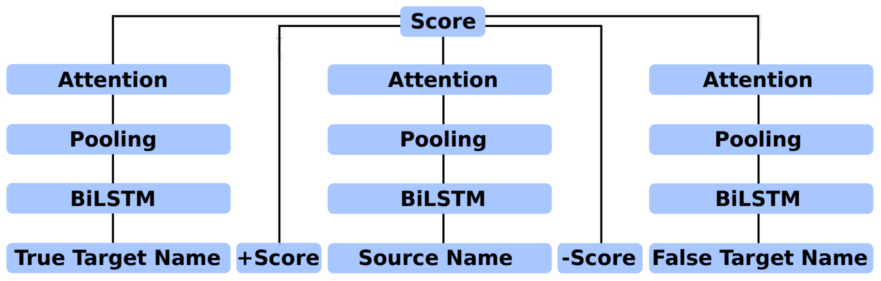

Fig. 1: The system architecture. The loss accepts two scores and three vectors to compute the difference between true and false distance given a source name.

True Target Name, Source Name, and False Target Name are character-level embedding layers for true, source, and false entity inputs. In the preprocessing step, +Score is computed as the cosine of the angle between source and true name vectors. Similarly,

-Score is the cosine of the angle between source and false name vectors. In this step, these two features are generated as tf-idf vectors of the most frequent n-grams of characters in their strings. The n-grams are limited to bi-, tri-, and four-grams.

As we observe in our experiments (see e.g. Table 2), these two scores have a significant impact on the performance of the network. For instance, take the source name “president Reagan” as the true match for “Ronald Reagan”. Simply depending on character representations would make “Nancy Reagan” a good match for “president Reagan”, too, which would be wrong. However, injecting the high cosine similarity of president Reagan and Ronald Reagan into the model as a signal instructs it to weigh the importance of similarities in a more elaborate way.

Character-level tf-idf vectors of source, true and false names are used as the inputs to the next layer, the BiLSTM modules for computing the string-level representations over which a column-wise max-pooling layer is applied. Then, an attention layer similar to [30] is used to help the model concentrate on more discriminating features.

The resulting vectors of the attention layers, as well as the scores, are inputs to the loss function (Equation 1). The loss function decreases the cost when the vectors of true matches get similar to the vectors of ground truths and vice versa.

$$\begin{aligned}\mathcal{L} = \max\{0, \mathbf{m} - \mathbf{{+}Score} * \mathscr{F}(\mathbf{S}, \mathbf{T}^{+}) \\ + \mathbf{{-}Score} * \mathscr{F}(\mathbf{S}, \mathbf{T}^{-})\}\end{aligned} \tag{1}$$

$$\mathscr{F} = \frac{1}{1 + \exp\left(-(\mathbf{v1} \cdot \mathbf{v2})\right)} * \frac{1}{1 + \|\mathbf{v1}, \mathbf{v2}\|} \tag{2}$$

Computing the similarity between two string at test time is done simply by using the same network parameters to represent the source and all formal names and computing their pair-wise similarities using $\mathscr{F}$ function (Equation 2). However, computing the $\mathscr{F}$ for all possible permutations of source vectors and formal vectors is infeasible when one or both of the lists are big. Our strategy for scaling up the NED is to use a window of the highest cosine scored formal names instead of using all of them. Our experiment on 1000 random samples with unlimited and limited formal names showed that the difference in none of the datasets is statistically significant (Table 2).

4 Experimental Results

We trained and evaluated our system on four publicly available datasets [27] compiled for alias detection. We refer to the datasets as Wiki, Wiki-people, Artists, and Patent Assignee. The Wiki dataset is compiled by assigning the hyperlinked string in Wikipedia pages to the page they are pointing to, assuming that Wikipedia pages are entities. The Wiki-people dataset is a subset of Wiki, which contains only entities with the type “person” in the Freebase [5] knowledge graph. The Artists dataset contains alternative names for music artists extracted from MusicBrains [26]. Finally, the Patent Assignee dataset contains the aliases of assignees in patent documents[2]. Table 1 displays some statistics of these datasets.

[2] There is a fifth dataset called “disease” which is compiled by the authors of [27]. This dataset was not publicly available at the time of authoring this work.

Dataset	Strings	Entities	Mentions	Train	Val.	Test
Wiki-people	1880000	1160000	1.83	51842	298	3946
Wiki	9320000	4640000	2.54	64341	288	3802
Artists	1830000	1160000	1.69	11566	265	3665
Patent Assignee	330000	227000	1.50	14365	290	3746

Table 1: Number of strings, number of entities, the average number of mentions per entity, and number of samples in the train, validation and test sets

All entities in the training data including true and false names are used to generate a list of most frequent n-gram characters limited to bi-, tri-, and four-grams. The list is used to encode the strings into their tf-idf feature vectors. The feature vectors are used for computing the cosine similarity, which is used as a feature in the neural network as well as a means to generate windows of false entities. False entities are sampled either randomly or from the window with the highest cosine scores. As an ablation study, several window sizes for false entities are selected to assess the impact of increasing false samples on the system performance.

We use BiLSTM modules with 128 units with Adam [13] as the optimizer and all dropouts set to 0.5. Since some source names may have more than one true and false answers, the Mean Average Precision (MAP) is used as the evaluation metric. We compare our system with two baselines, namely the plain Levenshtein and Jaro-Winkler distances and a state-of-the-art alias detection system proposed by [27]. The results of these experiments are reported in Table 2. The results show that our system outperforms the baselines by a large margin on three out of four datasets.

5 Ablation

To investigate different aspects of the system we performed an ablation study on several components of the system with the following variations. CW stands for Current Work.

– **CW-XN-ordered**
  To assess the impact of the window size or the number of false samples per true one, we define three window sizes shown as 1N, 2N, and 5N in Table 2. For instance 'CW-5N-ordered' means that for each true name we include 5 false names to train the system. At the inference time, there is no constraint on the number of false or true entities.
– **CW-2N-random**
  False names are selected either randomly or from a list of highest similar names. We make sure that the list does not contain any true target name. The distinction between these two experiments is shown by the suffix '-random' or '-ordered', respectively. The similarity scores used in this experiment are computed in the preprocessing step and are the same scores as used as a feature for training the model.
– **CW-2N-no-score**
  An additional experiment is conducted by removing the scores from the objective function to show the gain of this parameter in the network performance.

- **CW-2N-cosine**
  An experiment is conducted to assess the difference on the system performance by replacing the GESD [8] as the $\mathscr{F}$ in the objective function with cosine.
- **CW-2N-full-target**
  Finally, an experiment is performed to observe the impact of our filtering strategy on system performance. To make this experiment timely feasible, we randomly selected 1000 test samples and used the best performing model to disambiguate the samples. The results should be compared to the CW-2N-ordered experiment, which has exactly the same configuration but is applied on a limited window of 20 best scored formal names.

Model \ Dataset	Wiki	Wiki-people	Artists	Patent Assignee
Levenshtein	23.8	24.6	29.6	72.0
Jaro-Winkler	29.7	28.3	32.8	85.0
Tam et al. [27]	41.6	59.4	59.7	90.6
CW-1N-ordered	61.4	71.1	70.2	88.9
CW-2N-ordered	**61.7**	**71.3**	**70.4**	89.7
CW-5N-ordered	61.5	71.2	70.1	88.6
CW-2N-random	57.2	69.3	68.4	86.3
CW-2N-no-score	56.4	65.3	67.2	84.8
CW-2N-cosine	60.5	70.4	69.9	87.1
CW-2N-full-target	**61.8**	**71.2**	**70.5**	89.4

Table 2: The baselines and the results of several experiments conducted on different configurations of this work are reported using the Mean of Average Precision (MAP) metric. CW stands for Current Work. All models except CW-2N-cosine use GESD [8] for $\mathscr{F}$. Scores are all in percent.

As Table 2 shows, all variants of CW-1N-ordered, CW-2N-ordered, and CW-5N-ordered perform on par with each other, while CW-2N-ordered yields the best results. The gap between CW-2N-no-score and CW-2N-ordered signifies the importance of integrating source similarity scores as a soft constraint in the objective function.

Although the gap between CW-2N-full-target and CW-2N-ordered is not noticeable, the first model requires much more time at the inference step since it computes the similarity for all formal names while CW-2N-ordered computes it only for a limited number of formal names. This strategy for filtering formal names is crucial to make the disambiguation feasible when the number of formal names exceeds several thousand names. This experiment shows that the introduced strategy is effective to make large scale NED manageable while getting the same performance.

Comparing the results on CW-2N-ordered versus CW-2N-random shows that choosing false samples from instances with high similarity with the ground truth names enhances the performance of the model. Finally, compared to its counterpart with GESD, CW-2N-cosine performs poorly which suggests that better similarity functions can improve the network even more.

6 Conclusion

NED is an integral component in many NLP applications such as record linking, entity linking, or relation extraction. Large scale NED is particularly challenging due to the time it takes to extract the correct match among hundreds of thousands of formal names, given each source name.

We proposed a state-of-the-art system for large-scale NED. Our system consists of a deep architecture for pair-wise candidate ranking and a filtering scheme that allows the network to scale up to hundreds of thousands of formal names. We tested our system on four publicly available datasets and obtained superior results with large margins on three of them.

Ideally, including contextual data should improve the performance of a NED system. However, since neither of our datasets contains contextual data there is no way to assess the impact of providing contextual data on the system performance. Nevertheless, the proposed architecture is capable of modeling contextual data by concatenating them with its input vectors. In our future work, we would like to integrate formal names metadata as well as their surrounding context into the model to further improve the performance.

References

1. Aghaebrahimian, A.: Deep neural networks at the service of multilingual parallel sentence extraction. In: Proceedings of the International Conference on Computational Linguistics (CoLing). pp. 1372–1383 (2018)
2. Aghaebrahimian, A., Cieliebak, M.: Towards integration of statistical hypothesis tests into deep neural networks. In: Proceedings of the Association for Computational Linguistics (ACL). pp. 5551–5557 (2019)
3. Bergroth, L., Hakonen, H., Raita, T.: A survey of longest common subsequence algorithms. In: Proceedings Seventh International Symposium on String Processing and Information Retrieval (SPIRE). pp. 39–48 (2000)
4. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining. pp. 39–48 (2003)
5. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the conference of ACM Special Interest Group on Management of Data (SIGMOD). pp. 1247–1250 (2008)
6. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string metrics for matching names and records. In: Proceedings of the Workshop on Data Cleaning and Object Consolidation (2003)
7. Das, D., Smith, N.A.: Paraphrase identification as probabilistic quasi-synchronous recognition. In: Proceedings of the Association for Computational Linguistics (ACL). pp. 468–476 (2009)
8. Feng, M., Xiang, B., Glass, M.R., Wang, L., Zhou, B.: Applying deep learning to answer selection: A study and an open task. In: Proceedings of the workshop of Automatic Speech Recognition and Understanding (ASRU). pp. 33–40 (2015)
9. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: Proceedings of the Association for Computational Linguistics (ACL). pp. 363–370 (2005)
10. Gan, Z., Singh, P., Joshi, A., He, X., Chen, J., Gao, J., Deng, l.: Character-level deep conflation for business data analytics. arXiv:1702.02640 (2017)

11. Güngör, O., Üsküdarli, S., Güngör, T.: Improving named entity recognition by jointly learning to disambiguate morphological tags. In: Proceedings of the International Conference on Computational Linguistics (CoLing). pp. 2082–2092 (2018)
12. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence. pp. 2741–2749 (2016)
13. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
14. Kolitsas, N., Ganea, O.E., Hofmann, T.: End-to-end neural entity linking. In: Proceedings of the conference on Computational Natural Language Learning (CoNLL). pp. 519–529 (2018)
15. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 282–289 (2001)
16. Le, P., Titov, I.: Improving entity linking by modeling latent relations between mentions. In: Proceedings of the Association for Computational Linguistics (ACL). pp. 1595–1604 (2018)
17. Li, P., Dong, X.L., Guo, S., Maurino, A., Srivastava, D.: Robust group linkage. In: Proceedings of the 24th International Conference on World Wide Web. pp. 647–657 (2015)
18. Liu, L., Shang, J., Xu, F.F., Ren, X., Gui, H., Peng, J., Han, J.: Empower sequence labeling with task-aware neural language model. arXiv:1709.04109 (2017)
19. Luo, G., Huang, X., Lin, C.Y., Nie, Z.: Joint entity recognition and disambiguation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 879–888 (2015)
20. Madnani, N., Tetreault, J., Chodorow, M.: Re-examining machine translation metrics for paraphrase identification. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). pp. 182–190 (2012)
21. McCallum, A., Bellare, K., Pereira, F.: A conditional random field for discriminatively-trained finite-state string edit distance. In: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence. pp. 388–395 (2005)
22. Santos, R., Murrieta-Flores, P., Calado, P., Martins, B.: Toponym matching through deep neural networks. International Journal of Geographical Information Science **32**, 1–25 (2017)
23. Sil, A., Yates, A.: Re-ranking for joint named-entity recognition and linking. In: Proceedings of the International Conference on Information & Knowledge Management (CIKM). pp. 2369–2374 (2013)
24. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology **147**, 195 – 197 (1981)
25. Sultan, M.A., Bethard, S., Sumner, T.: DLS@CU: Sentence similarity from word alignment and semantic vector composition. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). pp. 148–153 (2015)
26. Swartz, A.: Musicbrainz: a semantic web service. IEEE Intelligent Systems **17**, 76–77 (2002)
27. Tam, D., Monath, N., Kobren, A., Traylor, A., Das, R., McCallum, A.: Optimal transport-based alignment of learned character representations for string similarity. In: Proceedings of the Association for Computational Linguistics (ACL). pp. 5907–5917 (2019)
28. Winkler, W.: The state of record linkage and current research problems. Statist. Med. (1999)
29. Yamada, I., Shindo, H., Takeda, H., Takefuji, Y.: Joint learning of the embedding of words and entities for named entity disambiguation. In: Proceedings of the Conference on Computational Natural Language Learning (CoNLL). pp. 250–259 (2016)
30. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL). pp. 1480–1489 (2016)