New Model, Who 'Dis?: Named Entity Linking using BERT and USE Contextualized Embeddings

Andrew Webb
School of Information, University of California, Berkeley
webban@berkeley.edu
December 7th, 2019

Abstract

Named Entity Linking (EL) is the task of identifying an entity within a natural language sequence and correctly linking that entity to the appropriate informational space in a Knowledge Base (KB). Previously successful attempts at EL required engineering extra features, such as adding entity co-occurrence statistics from a large text corpus to the KB[1], enriching the entity information with a predicted entity "type" (e.g. Person vs Place)[2], or creating a probabilistic map between entities and mentions[3]. It is the goal of this paper to see whether results comparable with current SOTA methods can be attained by using neural embeddings of the input text and KB entity summary.

While vector similarity proved to be problematic with BERT embeddings, Google's Universal Sentence Encoder allowed mention sequences to be compared to the Wikipedia summary for the task of Entity Disambiguation. Additionally, it was found that the embeddings alone are not quite sufficient for competitive accuracies. However, leveraging existing search functionality in Wikipedia (and likely other KBs) proved to be an effective and practical, though perhaps not fully valid in the academic sense, approach.

Introduction and Motivation

With the accessibility of KBs across both public (e.g. Wikipedia) and private (e.g. LinkedIn) domains, there is a wealth of opportunity to connect the abundant amount of real-world data from news articles, social media, email exchanges, and other digital sources to information in a KB. Accurately identifying entities from real-world mentions and descriptions and linking them back to a KB can help organizations in a variety of ways, including, but not limited to:

Understanding entity relationships
"Deutsche Telekom eyes potential merger with Orange – report"[4]

Identifying commercial opportunities
"Netflix is due for good news and 'The Irishman' could be the answer, says JP Morgan"[5]

Gathering sentiment
"Disney's cringe-worthy Baby Yoda merch goes on sale"[6]

Stated formally, the task of EL consists of taking inputs from a corpus of text *T*, which has a set of entity mentions *M*, and linking each entity mention to the corresponding entity in the set of entities *E* within the KB. In order to link an entity mentioned in a natural language input to the correct space within the KB, there are generally three main tasks that need to be completed: 1) Named Entity Recognition, in which the Named Entity is distinguished from non-entity tokens, 2) Entity Candidate Generation and Selection, in which *N* most likely candidates from the KB are selected as possible matches to the identified entity, and 3) Un-linkable mention prediction, in which the identified entity is acknowledged to have no existing information in the KB.[4]

The goal of the research summarized by this paper is to show that even novice NLP practitioners can accomplish each of the

complex tasks necessary for entity-linking using a relatively simple, yet effective, model.

The CoNLL-AIDA dataset is deemed as the most practical input text for training due to its simplicity and scale (1,392 total documents, with ~37.6 thousand annotated entity mentions). Using BERT and USE, both newly developed tools for contextualized sequence embeddings, the mention text and the entity summary pages in the KB are embedded as vectors. The embeddings not only help with the initial Named Entity Recognition task, but also the downstream tasks of Entity Candidate Generation/Selection and Un-linkable mention prediction. Once entities are identified in the neural network, the cosine similarity of the entity mention with some surrounding context and the KB summary page helps find the most probable entity linkage among possible KB candidates.

Related Work

Many of the more effective EL methods require engineered features. One common method is generating Entity Name Dictionaries which take a mention text (Obama, BO, Barack) as keys and map it to a corresponding entity value (Barack Obama)[7]. A knowledge base such as Wikipedia facilitates the creation of these dictionaries as entity, redirect, and disambiguation pages can all be leveraged to build the key-value objects. Matches or partial matches to the mention text key can help a system identify entities within other text as well as linking to the appropriate value corresponding to the correct entity. However, not only is the task of generating the dictionary laborious, but it is possible that the system becomes too dependent on the dictionary keys for NER and EL, leading to low levels of generalization.

More current approaches include leveraging predictive models to create features to aid the EL task. A current SOTA method for disambiguating entities (a crucial component of accurate EL) predicts the 'type' of entity using the context around the entity[2]. This derived entity type enables the model to correctly differentiate (and therefore accurately link) between entities such as Washington (the place) and Washington (the person). While effective, this type system is fairly complex and is created completely outside of the input text and KB data. Ideally, accurate EL could be achieved with minimally derived features.

One such approach has been achieved by Kolitsas, Ganea, and Hofmann[8], who replaced all engineered features in favor of learned neural entity embeddings from the input text. While previous researchers have used contextual word and entity embeddings for the task of entity disambiguation[3], Kolitsas, Ganea, and Hofmann utilize Word2Vec word embeddings with a bidirectional LSTM to both identify entities within a span of text, and then find the most probable KB entity for linking using the word embeddings, w, with pre-trained entity embeddings, e, such that the probability of an entity given the word embeddings, $\hat{p}(w|e)$, is maximized. Further, they add a final shallow feed-forward neural network to "globally" disambiguate the entities within an entire document. This model achieves the current SOTA metrics for end-to-end (as opposed to entity disambiguation only) EL approaches.

Taking inspiration from the work of Kolitsas, Ganea, and Hoffman, the research summarized in this paper aims to utilize pre-trained word embeddings to create an even simpler, yet still effective, model for end-to-end EL. Specifically, this model uses BERT token embeddings to first identify the entity, and then Universal Sentence Encoder embeddings to find which summary from a set of entity candidates has the closest similarity to the mention text. Therefore, there is no offline or prior work required to create an empirical probabilistic entity map. The power and ease-of-use of the BERT/USE embeddings allow for the creation of a very practical, if not SOTA, model.

Methodology

The entire methodology for this project is centered around getting accurate results with practical approaches. The ideal outcome would

not require outside work, extensive memory and/or compute hardware, or deep expertise. Evaluation of success or failure of any approach is defined by the accuracy of the predicted linking versus the actual linking in the validation and test datasets.

As mentioned previously, the simplicity and size of the CoNLL-AIDA dataset lends itself particularly useful to the overall goal of a simple and practical model. The CoNLL-AIDA dataset does not identify entities with the traditional supplementary NER tags, such as PER, ORG, GEO, etc. One can reasonably assume that it is therefore more in-line with real-world datasets which may be compiled to identify entities but not deal with the extra task of entity type. Additionally, each entity mention is annotated with the corresponding entity name, or names, and Wikipedia URL.

With some preprocessing, the CoNLL-AIDA dataset is formatted such that it can be consumed by a BERT neural net input layer. Using the sequence output, each token is represented as a 768-dimensional vector before being further processed by 2 sets of a dense neural layer followed by a dropout layer to help generalize to the validation and test dataset. A softmax-activated layer serves as the model's entity recognition tool to predict if the text token is either the entity beginning/middle or an 'other' token, i.e. not an entity. Sparse Categorical Cross Entropy is the loss function used as most of the tokens will be represented by 'Other' NER tags. In addition to standard accuracy, the NER model is evaluated with a custom accuracy function which considers only the non-Other entity tags to show how accurate it is at getting the entity predictions correct. Accurate 'is-entity' or 'is-not-entity' predictions are crucial for the downstream entity disambiguation and linking tasks, so several experiments, detailed in the following section, are performed on the model's dropout rate and custom Adam optimizer parameters to get the best entity recognition results.

With entities accurately identified, the task becomes correctly disambiguating the entity and linking it to the corresponding Wikipedia page. To do this, the Wikipedia summary is used to differentiate entities (i.e. distinguish between Phil Simmons the cricket player and Phil Simmons the Olympian rower). Text surrounding the identified entity is then used to give context around the mention. How many characters to use from the Wikipedia summary and the surrounding mention text are parameters that can be adjusted to boost accuracy (details in the following section).

Several methods were attempted to capture text from Wikipedia summaries and embed it as a vector. The last method proved to be the most effective and practical.

1. Download the summary text for all Wikipedia entities. This would require significant amounts of memory in order to compute vector similarity between the summary and mention text, which would likely have to be done in many batches on any standard machine
2. As a proof-of-concept, "cheat" a little bit: take each entity in the training, validation, and test dataset and download an additional set of summaries using Wikipedia search functionality. This decreases the amount of memory and processing needed while maintaining the essence of the disambiguation task by creating sets of similar entities
3. Building upon the previous method, utilize the Wikipedia search functionality for each entity identified by the NER model. Embed the summaries as vectors and choose the entity based on the vector similarity with the mention. This further reduces the amount of memory and processing required and is a more practical method as it is not necessary to store many entity embeddings. Search functionality is not unique to Wikipedia, therefore this method can be applied outside of the context of this research as well

For each of these methods, embedding the mention text and the entity summary was initially done using BERT. This did not give optimal results however, as the cosine similarity

between the mention text and the entity candidates was usually very low (<0.05 for most cases, indicating that the vectors are nearly orthogonal and the text is unrelated). This is possibly due to the high-dimensional nature of the BERT embeddings. Even when using high weight scores to give more attention to the entity within the mention text (e.g. weighting the underlined tokens 100x more than the rest in the sentence: ..trade with Poland rose to $143.3 million..), the cosine similarity was still very low. For this reason, Google's Universal Sentence Encoder was used with much more desirable results (mention text and the correct entity summary embeddings often had the highest cosine similarity for all candidates, even when candidates had the same name).

To evaluate the effectiveness of this approach, the below baseline measure is used:

Baseline – Take the text for the identified entity and just use that to create the Wikipedia URL (e.g. when the NER model identifies "Leicestershire" as an entity, just add that text as a suffix to the standard Wikipedia URL – 'http://en.wikipedia.org/wiki/Leicestershire'). This has obvious disadvantages, for example the text "Leicestershire 11 points, Hampshire 7" refers not to Leicestershire the place, but rather the cricket club, which has a different URL. This baseline approach gives the correct entity linkage on ~82.3% of the validation dataset.

Model performance against the baseline is described in the 'Results' section.

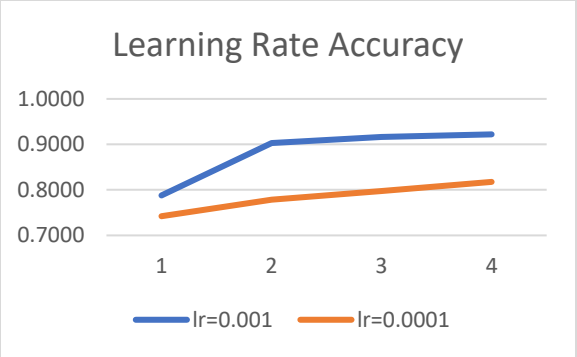
Experiments

Experiments have been divided into two sections, those that were performed to get the best accuracy identifying entities, and those that were performed to later get the best accuracy linking the identified entities to the KB. The below experiments were run using only 1000 training and validation sentences.

Named Entity Recognition Experiments

Initial NER model builds did a good job of accurately predicting entities on the training set but did not generalize well to the validation and test set. Dropout layers were therefore added to the model and the dropout rate was a hyperparameter that was adjusted to influence the accuracy outside of the training set. Appendix Table 1 illustrates the average metric values for the last epoch after running the model 3 times. The average is used to account for the random variations in model optimization and initialization, which slightly vary metric values. The best 'is_entity_accuracy' on the reduced validation dataset is achieved by setting the first dropout rate to 0.3 and the second dropout rate to 0.1. Dropping a higher proportion of the neural net elements earlier seems to help the model better generalize to unseen data. Interestingly, mixing the dropout rates gave better performance than having the two dropout layers have the same rates (though when the dropout rate was 0.1 for both layers it was close to the mixed-rate results).

Next, custom parameters for the model's Adam optimizer were adjusted to further increase 'is_entity_accuracy'. Using a standard Adam optimizer generally lead to gradient issues which caused the model accuracy to spiral downwards, often happening in the first epoch. Deviating only slightly from the standard parameter values resulted in significantly more accurate, and reliable results (i.e. no instances of accuracy dropping to zero during training). Experiment details are shown in Appendix Table 2, which also shows average metric values after running the model several times with the same custom parameter values. Reducing the learning rate prevents the model from spiraling too far in the wrong direction when attempting to minimize the loss, however if only the learning rate is reduced, the model accuracy plateaus very early at ~80% after 4 epochs as illustrated in the below figure. Accuracy can be improved using the lower learning rate, however many more epochs are needed.



To address this, the level of decay is also experimented with so that the model can start with a higher learning rate which will decrease as the model learns. As the model runs through more batches and epochs, and as the accuracy improves during training, the learning rate will decrease to avoid large steps in the wrong direction during optimization. On the reduced validation dataset, the greatest average 'is_entity_accuracy' was achieved with a learning rate of 0.0005, beta 1 of 0.915, and decay of 0.05.

Entity Disambiguation/Linking Experiments

In order to find vector similarity between the mention text and the entity summary, the amount of text from both the mention and entity summary can be adjusted to find what yields the greatest accuracy respective to disambiguating, and then linking, the entity (measured as how accurate the predicted Wikipedia URL is against the actual URL).

For mention text, *k* number of tokens before and after the identified entity tokens are embedded using the USE. For example, in the text: "…advice to consumers to shun British lamb until scientists determine whether…", if *k* is set to 3, the mention sequence would be: "consumers to shun British lamb until scientists". Experiments detailed in Appendix Table 3 show the linking accuracy when *k* values are set to 1, 3, 5, and 7.

While setting *k* did not significantly impact results, 3 and 7 surrounding tokens resulted in the highest accuracies on the validation dataset. 3 was used in the final model since the accuracies for both were identical.

For entity summaries, the number of characters from the summary was set to 25, 50, 100, and 200 before embedding the text as a vector using USE. Details are shown in Appendix Table 4. Using less characters from the Wikipedia summary yielded higher accuracy on the validation dataset.

Results

When the finalized dropout rates and Adam parameters were applied to the entire training dataset, the NER model achieved, on average after several runs, 'is_entity_accuracy' of ~96.5% on the validation dataset, and ~95.6% on the test dataset. Having a high amount of accuracy relative to the entity identification enabled the next step of disambiguation and linking. Not only does the model achieve a high level of accuracy, but it also is run with only 8 epochs and finishes in roughly ~40 minutes on a standard machine equipped with a GPU.

Using the optimal surrounding text and Wikipedia summary character length parameters, the model is able to correctly predict the entity linkage on ~84.6% of the entity mentions in the test dataset. This outperforms the baseline.

While the EL is accurate, it isn't perfect. The model still has challenges disambiguating entities that are very similar (e.g. the token 'Heineken' is linked to the 'Heineken' Wikipedia page by the model whereas the correct entity linkage is to 'Heineken Pilsner'). This could possibly be addressed by using weights to give more influence to the entity mention within the surrounding text sequence (more details on that in the following section).

Next Steps

While similarity measures between mention and entity summary embeddings have been shown to produce accurate EL results, another area of interest is using the Wikipedia search functionality within the neural network during training time. Search results from training dataset entities can be used as negative examples against the positive/true entities. This would

effectively consolidate the NER and EL task within the same neural network. This should also improve accuracy for ambiguous entities that are mentioned multiple times, either within the same document or even in different documents, since the BERT sequence embeddings would be paired with positive entity linking examples. At inference time, the model could predict if the sequence text is valid entity, extract *N* potential candidates, and then predict the most probable candidate for disambiguation and linkage. The risk of this approach is that the extra time taken to search for entity candidates can impact prediction time.

One of the areas of the model's EL inaccuracy has to do with entities that in the CoNLL-AIDA dataset that are not linked to a valid Wikipedia page. Using Wikipedia's search functionality always pulls *N* most probably entities given the input text, therefore the model rarely predicts that an entity is an un-linkable mention. The Wikipedia search function also pulls the most current data, meaning that potentially an entity that did not have a Wikipedia page when the CoNLL-AIDA dataset was created now has a page and that page is used for the predicted linkage. To address this, vector similarity values below a certain threshold could be marked as un-linkable mentions. This threshold could even be learned by a machine-learning model built to optimize the accuracy of un-linkable mention predictions. This isn't really a high priority however, as practical usage of EL models will most likely use real-time information rather than a dataset that can possibly become outdated.

The model also has some issues with casing. For example, LEICESTERSHIRE and Leicestershire are the same entity, however the BERT tokenizer tokenizes them differently. This leads to inaccurate entity recognition especially for entities in article titles as those are often all capitalized. Additional work could be necessary to tackle this problem in the pre-processing stage to increase NER model accuracy.

Lastly, the USE does not appear to allow weights to be used when pooling sequences of text together to create a sentence embedding. Some sort of weighting mechanism should help the accuracy of the model as one could give higher weights to the mention text for the entity and lower weights to the surrounding context (which is still needed to differentiate similar entities such as 'Madrid' the city and 'Real Madrid' the football club). Perhaps using BERT sequence output embeddings, the vectors could be properly weighted to give better accuracy.

Conclusion

The methods discussed in this paper outline simple, practical, and effective approaches to EL that anyone can attain with relatively simple hardware, foundational knowledge of NLP, and a few hours to spare.

The end-to-end EL framework accurately identifies and disambiguates entities (with accurate disambiguation you get the linking as a bonus since the Wikipedia URL is composed from the entity name) without requiring complex probabilistic mappings or other features created offline.

The Wikipedia search functionality significantly helps the task of entity disambiguation. While this search functionality uses features outside of the model and may feel a bit like cheating, it is far more practical than storing hundreds of thousands, or even millions, of entity summary embeddings to disambiguate identified entities.

While this approach may not follow traditional academic approaches to EL, it can still be used by NLP practitioners on various datasets to link to a KB of interest as long as that KB has some sort of search functionality (which many do).

Appendix

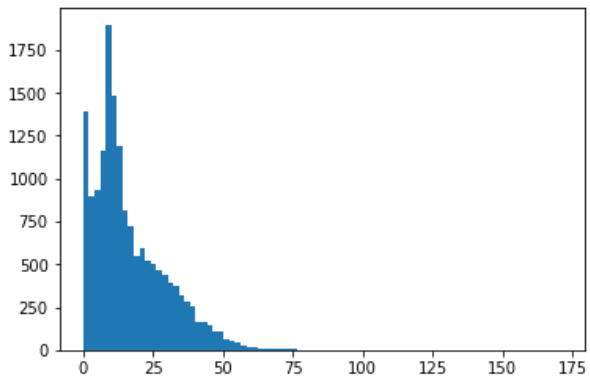
References

1. Priya Radhakrishnan, Partha Talukdar, Vasudeva Varma 2018. *ELDEN: Improved Entity Linking using Densified Knowledge Graphs* https://www.aclweb.org/anthology/N18-1167.pdf
2. Jonathan Raiman, Olivier Raiman 2018. *Deep Type: Multilingual Entity Linking by Neural Type System Evolution* https://arxiv.org/pdf/1802.01021.pdf
3. Nikolaos Kolitsas, Octavian-Eugen Ganea 2018. *End-to-End Neural Entity Linking* https://arxiv.org/pdf/1808.07699.pdf
4. Seeking Alpha 2019. *Deutsche Telekom eyes potential merger with Orange - report* https://seekingalpha.com/news/3522389-deutsche-telekom-eyes-potential-merger-orange-report
5. CNBC 2019. *Netflix is due for good news and 'The Irishman' could be the answer, says JP Morgan* https://www.cnbc.com/2019/11/27/netflix-is-due-for-good-news-and-the-irishman-could-be-the-answer-says-jp-morgan.html
6. TechCrunch 2019. *Disney's cringe-worthy Baby Yoda merch goes on sale* https://techcrunch.com/2019/11/26/baby-yoda-merchandise/
7. Abhishek Gattani, Digvijay S. Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman , Venky Harinarayan, AnHai Doan 2013. *Entity Extraction, Linking, Classification, and Tagging for Social Media: A Wikipedia-Based Approach* http://www.mitultiwari.net/docs/papers/vldb_social_categorization.pdf

Code for all of the methods and results described in this paper can be found at the below public Github repo

https://github.com/webban0014/w266_project

Training dataset sentence length distribution



Example NER model training/validation loss, accuracy, and is_entity_accuracy values

Epoch 1/8

16108/16108 [==============================] - 277s 17ms/sample - loss: 0.0307 - acc: 0.9899 - is_entity_accuracy: 0.9149 - val_loss: 0.0179 - val_acc: 0.9942 - val_is_entity_accuracy: 0.9493

Epoch 2/8

16108/16108 [==============================] - 300s 19ms/sample - loss: 0.0136 - acc: 0.9958 - is_entity_accuracy: 0.9660 - val_loss: 0.0161 - val_acc: 0.9951 - val_is_entity_accuracy: 0.9578

Epoch 3/8

16108/16108 [==============================] - 314s 20ms/sample - loss: 0.0103 - acc: 0.9970 - is_entity_accuracy: 0.9749 - val_loss: 0.0157 - val_acc: 0.9954 - val_is_entity_accuracy: 0.9621

Epoch 4/8

16108/16108 [==============================] - 321s 20ms/sample - loss: 0.0085 - acc: 0.9975 - is_entity_accuracy: 0.9803 - val_loss: 0.0165 - val_acc: 0.9954 - val_is_entity_accuracy: 0.9586

Epoch 5/8

16108/16108 [==============================] - 324s 20ms/sample - loss: 0.0072 - acc: 0.9979 - is_entity_accuracy: 0.9835 - val_loss: 0.0161 - val_acc: 0.9955 - val_is_entity_accuracy: 0.9654

Epoch 6/8

16108/16108 [==============================] - 329s 20ms/sample - loss: 0.0062 - acc: 0.9982 - is_entity_accuracy: 0.9867 - val_loss: 0.0169 - val_acc: 0.9956 - val_is_entity_accuracy: 0.9654

Epoch 7/8

16108/16108 [==============================] - 329s 20ms/sample - loss: 0.0054 - acc: 0.9985 - is_entity_accuracy: 0.9889 - val_loss: 0.0173 - val_acc: 0.9956 - val_is_entity_accuracy: 0.9647

Epoch 8/8

16108/16108 [==============================] - 330s 20ms/sample - loss: 0.0048 - acc: 0.9986 - is_entity_accuracy: 0.9896 - val_loss: 0.0177 - val_acc: 0.9955 - val_is_entity_accuracy: 0.9661

Example text sentence with ‘is_entity_prediction’ value compared against actual value

Text	is_entity_prediction	True NER tag
'Japan'	1	1
'began'	0	0
'the'	0	0
'defence'	0	0
'of'	0	0
'their'	0	0
'Asian'	1	1
'Cup'	2	2
'title'	0	0
'with'	0	0
'a'	0	0
'lucky'	0	0
'2'	0	0
'-'	0	0
'1'	0	0
'win'	0	0
'against'	0	0
'Syria'	1	1
'in'	0	0
'a'	0	0
'Group'	0	0
'C'	0	0
'championship'	0	0
'match'	0	0
'on'	0	0
'Friday'	0	0
'.'	0	0

Table 1 – Dropout parameter effects on training and validation accuracy and loss

			training dataset			validation dataset		
	1st DO layter	2nd DO layer	loss	acc	is_entity_accuracy	val_loss	val_acc	val_is_entity_accuracy
dropoutRate	0.5	0.5	0.4516	0.8917	0.1691	0.4503	0.9101	0.0000
	0.3	0.3	0.2004	0.9723	0.7691	0.1023	0.9753	0.7398
	0.2	0.2	0.4138	0.8970	0.0000	0.3639	0.9101	0.0000
	0.1	0.1	0.0408	0.9871	0.9003	0.0542	0.9838	0.8393
	0.3	0.1	0.0366	0.9885	0.9085	0.0434	0.9888	0.8931
	0.1	0.3	0.0421	0.9871	0.9045	0.0565	0.9833	0.8274

Table 2 – Custom Adam optimizer parameter effects on training and validation accuracy and loss

			training dataset			validation dataset		
custom adam params			loss	acc	is_entity_accuracy	val_loss	val_acc	val_is_entity_accuracy
lr=0.001	beta_1=0.91	decay=0.1	0.0130	0.9959	0.9722	0.0376	0.9908	0.9221
lr=0.0001	beta_1=0.91	decay=0.1	0.0610	0.9785	0.8241	0.0534	0.9816	0.8176
lr=0.001	beta_1=0.91	decay=0.05	0.0122	0.9963	0.9751	0.0402	0.9909	0.9137
lr=0.001	beta_1=0.92	decay=0.1	0.0191	0.9937	0.9515	0.0357	0.9903	0.9115
lr=0.0001	beta_1=0.91	decay=0.05	0.0164	0.9950	0.9610	0.0384	0.9887	0.9014
lr=0.0005	beta_1=0.91	decay=0.05	0.0015	0.9996	0.9968	0.0477	0.9918	0.9262
lr=0.0005	beta_1=0.92	decay=0.05	0.0011	0.9998	0.9981	0.0498	0.9916	0.9225
lr=0.0005	beta_1=0.915	decay=0.05	0.0013	0.9996	0.9970	0.0501	0.9913	0.9313

Table 3 – Surrounding sequence length parameter effects on validation dataset Entity Disambiguation/Linking accuracy

Surrounding Sequence Length	Validation EL accuracy
1	0.84625
3	0.84750
5	0.84625
7	0.84750

Table 4 – Number of Wikipedia summary characters effects on validation dataset Entity Disambiguation/Linking accuracy

# of Wikipedia Summary Characters	Validation EL accuracy
25	0.84875
50	0.84625
100	0.83500
200	0.84688