A Lightweight Neural Model for Biomedical Entity Linking

Lihu Chen,[1] **Gaël Varoquaux,**[2] **Fabian M. Suchanek,**[1]
[1] LTCI & Télécom Paris & Institut Polytechnique de Paris, France
[2] Inria & CEA & Université Paris-Saclay, France
lihu.chen@telecom-paris.fr, suchanek@telecom-paris.fr, gael.varoquaux@inria.fr

Abstract

Biomedical entity linking aims to map biomedical mentions, such as diseases and drugs, to standard entities in a given knowledge base. The specific challenge in this context is that the same biomedical entity can have a wide range of names, including synonyms, morphological variations, and names with different word orderings. Recently, BERT-based methods have advanced the state-of-the-art by allowing for rich representations of word sequences. However, they often have hundreds of millions of parameters and require heavy computing resources, which limits their applications in resource-limited scenarios. Here, we propose a lightweight neural method for biomedical entity linking, which needs just a fraction of the parameters of a BERT model and much less computing resources. Our method uses a simple alignment layer with attention mechanisms to capture the variations between mention and entity names. Yet, we show that our model is competitive with previous work on standard evaluation benchmarks.

1 Introduction

Entity linking (Entity Normalization) is the task of mapping entity mentions in text documents to standard entities in a given knowledge base. For example, the word "Paris" is *ambiguous*: It can refer either to the capital of France or to a hero of Greek mythology. Now given the text "Paris is the son of King Priam", the goal is to determine that, in this sentence, the word refers to the Greek hero, and to link the word to the corresponding entity in a knowledge base such as YAGO (Suchanek, Kasneci, and Weikum 2007) or DBpedia (Auer et al. 2007).

In the biomedical domain, entity linking maps mentions of diseases, drugs, and measures to normalized entities in standard vocabularies. It is an important ingredient for automation in medical practice, research, and public health. Different names of the same entities in Hospital Information Systems seriously hinder the integration and use of medical data. If a medication appears with different names, researchers cannot study its impact, and patients may erroneously be prescribed the same medication twice.

The particular challenge of biomedical entity linking is not the ambiguity: a word usually refers to only a single entity. Rather, the challenge is that the surface forms vary markedly, due to abbreviations, morphological variations, synonymous words, and different word orderings. For example, *"Diabetes Mellitus, Type 2"* is also written as *"DM2"* and *"lung cancer"* is also known as *"lung neoplasm malignant"*. In fact, the surface forms vary so much that all the possible expressions of an entity cannot be known upfront. This means that standard disambiguation systems cannot be applied in our scenario, because they assume that all forms of an entity are known.

One may think that variation in surface forms is not such a big problem, as long as all variations of an entity are sufficiently close to its canonical form. Yet, this is not the case. For example, the phrase *"decreases in hemoglobin"* could refer to at least 4 different entities in MedDRA, which all look alike: *"changes in hemoglobin"*, *"increase in hematocrit"*, *"haemoglobin decreased"*, and *"decreases in platelets"*. In addition, biomedical entity linking cannot rely on external resources such as alias tables, entity descriptions, or entity co-occurrence, which are often used in classical entity linking settings.

For this reason, entity linking approaches have been developed particularly for biomedical entity linking. Many methods use deep learning: the work of Li et al. (2017) casts biomedical entity linking as a ranking problem, leveraging convolutional neural networks (CNNs). More recently, the introduction of BERT has advanced the performance of many NLP tasks, including in the biomedical domain (Huang, Altosaar, and Ranganath 2019; Lee et al. 2020; Ji, Wei, and Xu 2020). BERT creates rich pre-trained representations on unlabeled data and achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures. However, considering the number of parameters of pre-trained BERT models, the improvements brought by fine-tuning them come with a heavy computational cost and memory footprint. This is a problem for energy efficiency, for smaller organizations, or in poorer countries.

In this paper, we introduce a very lightweight model that achieves a performance statistically indistinguishable from the state-of-the-art BERT-based models. The central idea is to use an alignment layer with an attention mechanism, which can capture the similarity and difference of corresponding parts between candidate and mention names. Our

model is 23x smaller and 6.4x faster than BERT-based models on average; and more than twice smaller and faster than the lightweight BERT models. Yet, as we show, our model achieves comparable performance on all standard benchmarks. Further, we can show that adding more complexity to our model is not necessary: the entity-mention priors, the context around the mention, or the coherence of extracted entities (as used, e.g., in Hoffart et al. 2011) do not improve the results any further. [1]

2 Related Work

In the biomedical domain, much early research focuses on capturing string similarity of mentions and entity names with rule-based systems (Dogan and Lu 2012; Kang et al. 2013; D'Souza and Ng 2015). Rule-based systems are simple and transparent, but researchers need to define rules manually, and these are bound to an application.

To avoid manual rules, machine-learning approaches learn suitable similarity measures between mentions and entity names automatically from training sets (Leaman, Islamaj Doğan, and Lu 2013; Doğan, Leaman, and Lu 2014; Ghiasvand and Kate 2014; Leaman and Lu 2016). However, one drawback of these methods is that they cannot recognize semantically related words.

Recently, deep learning methods have been successfully applied to different NLP tasks, based on pre-trained word embeddings, such as word2vec (Mikolov et al. 2013) and Glove (Pennington, Socher, and Manning 2014). Li et al. (2017) and Wright (2019) introduce a CNN and RNN, respectively, with pre-trained word embeddings, which casts biomedical entity linking into a ranking problem.

However, traditional methods for learning word embeddings allow for only a single context-independent representation of each word. Bidirectional Encoder Representations from Transformers (BERT) address this problem by pre-training deep bidirectional representations from unlabeled text, jointly conditioning on both the left and the right context in all layers. Ji, Wei, and Xu (2020) proposed an biomedical entity normalization architecture by fine-tuning the pre-trained BERT / BioBERT / ClinicalBERT models (Devlin et al. 2018; Huang, Altosaar, and Ranganath 2019; Lee et al. 2020). Extensive experiments show that their model outperforms previous methods and advanced the state-of-the-art for biomedical entity linking. A shortcoming of BERT is that it needs high-performance machines.

3 Our Approach

Formally, our inputs are (1) a *knowledge base* (KB), i.e., a list of entities, each with one or more names, and (2) a *corpus*, i.e., a set of text documents in which certain text spans have been tagged as entity mentions. The goal is to link each entity mention to the correct entity in the KB. To solve this problem, we are given a training set, i.e., a part of the corpus where the entity mentions have been linked already to the correct entities in the KB. Our method proceeds in 3 steps:

[1] All data and code are available at https://github.com/tigerchen52/Biomedical-Entity-Linking.

Preprocessing. We preprocess all mentions in the corpus and entity names in the KB to bring them to a uniform format.

Candidate Generation. For each mention, we generate a set of candidate entities from the KB.

Ranking Model. For each mention with its candidate entities, we use a ranking model to score each pair of mention and candidate, outputting the top-ranked result.

Let us now describe these steps in detail.

3.1 Preprocessing

We preprocess all mentions in the corpus and all entity names in the KB by the following steps:

Abbreviation Expansion. Like previous work (Ji, Wei, and Xu 2020), we use the Ab3p Toolkit (Sohn et al. 2008) to expand medical abbreviations. The Ab3p tool outputs a probability for each possible expansion, and we use the most probable expansion. For example, Ab3p knows that "DM" is an abbreviation of "Diabetes Mellitus", and so we replace the abbreviation with its expanded term. We also expand mentions by the first matching one from an abbreviation dictionary constructed by previous work (D'Souza and Ng 2015), and supplement 20 biomedical abbreviations manually (such as Glycated hemoglobin (HbA1c)). Our dictionary is available in the supplementary material and online.

Numeral Replacement. Entity names may contain numerals in different forms (e.g., Arabic, Roman, spelt out in English, etc.) We replace all forms with spelled-out English numerals. For example, "type II diabetes mellitus" becomes "type two diabetes mellitus". For this purpose, we manually compiled a dictionary of numerals from the corresponding Wikipedia pages. Finally, we remove all punctuation, and convert all words to lowercase.

KB Augmentation. We augment the KB by adding all names from the training set to the corresponding entities. For example, if the training set links the mention "GS" in the corpus to the entity "Adenomatous polyposis coli" in the KB, we add "GS" to the names of that entity in the KB.

3.2 Candidate Generation

Our ranking approach is based on a deep learning architecture that can compute a similarity score for each pair of a mention in the corpus and an entity name in the KB. However, it is too slow to apply this model to all combinations of all mentions and all entities. Therefore, we generate, for each mention M in the corpus, a set C_M of candidate entities from the KB. Then we apply the deep learning method only to the set C_M.

To generate the candidate set C_M, we calculate a score for M and each entity in the KB, and return the top-k entities with the highest score as the candidate set C_M (in our experiments, $k = 20$). As each entity has several names, we calculate the score of M and all names of the entity E, and use the maximum score as the score of M and the entity E.

To compute the score between a mention M and an entity name S, we split each of them into tokens, so that we have $M = \{m_1, m_2, ..., m_{|M|}\}$ and $S = \{s_1, s_2, ..., s_{|S|}\}$.

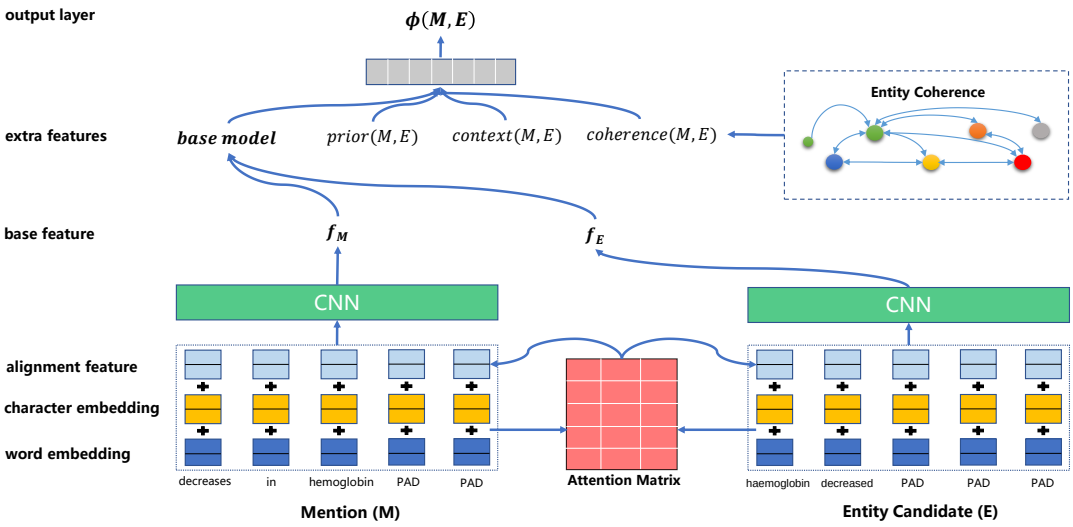

Figure 1: The architecture of our ranking model, with the input mention "decreases in hemoglobin" and the input entity candidate "haemoglobin decreased".

We represent each token by a vector taken from pre-trained embedding matrix $\mathbf{V} \in \mathbb{R}^{d \times |V|}$ where d is the dimension of word vectors and V is a fixed-sized vocabulary (details in Section 4.2). To take into account the possibility of different token orderings in M and S, we design the *aligned cosine similarity* (*ACos*), which maps a given token $m_i \in M$ to the most similar token $s_j \in S$ and returns the cosine similarity to that token:

$$ACos(m_i, S) = \max\{cos(m_i, s_j) \mid s_j \in S\} \quad (1)$$

The similarity score is then computed as the sum of the aligned cosine similarities. To avoid tending to long text, and to make the metric symmetric, we add the similarity scores in the other direction as well, yielding:

$$sim(M,S) = \frac{1}{|M|+|S|}\Big(\sum_{m_i \in M} ACos(m_i, S) + \sum_{s_j \in S} ACos(s_j, M)\Big) \quad (2)$$

We can now construct the candidate set $C_M = \{\langle E_1, S_1\rangle, \langle E_2, S_2\rangle, ..., \langle E_k, S_k\rangle\}$ where E_i is the id of the entity, and S_i is the chosen name of the entity. This set contains the top-k ranked entity candidates for each mention M. Specifically, if there are candidates whose score is equal to 1 in this set, we will filter out other candidates whose score is less than 1.

3.3 Ranking Model

Given a mention M and its candidate set $C_M = \{\langle E_1, S_1\rangle, \langle E_2, S_2\rangle, ..., \langle E_k, S_k\rangle\}$, the ranking model computes a score for each pair of the mention and an entity name candidate S_i. Figure 1 shows the corresponding neural network architecture. Let us first describe the base model. This model relies exclusively on the text similarity of mentions and entity names. It ignores the context in which a mention appears, or the prior probability of the target entities. To compute the text similarity, we crafted the neural network following the candidate generation: it determines, for each token in the mention, the most similar token in the entity name, and vice versa. Different from the candidate generation, we also take into account character level information here and use an alignment layer to capture the similarity and difference of correspondences between mention and entity names.

Representation Layer. As mentioned in Section 3.2, we represent a mention M and an entity name S by the set of the embeddings of its tokens in the vocabulary V. However, not all tokens exist in the vocabulary V. To handle out-of-vocabulary words, we adopt a recurrent Neural Network (RNN) to capture character-level features for each word. This has the additional advantage of learning the morphological variations of words. We use a Bi-directional LSTM (BiLSTM), running a forward and backward LSTM on a character sequence (Graves, Mohamed, and Hinton 2013). We concatenate the last output states of these two LSTMs as the character-level representation of a word. To use both word-level and character-level information, we represent each token of a mention or entity name as the concatenation of its embedding in V and its character-level representation.

Alignment Layer. To counter the problem of different word orderings in the mention and the entity name, we want the network to find, for each token in the mention, the most similar token in the entity name. For this purpose, we

adapt the attention mechanisms that have been developed for machine comprehension and answer selection (Chen et al. 2016; Wang and Jiang 2016).

Assume that we have a mention $M = \{\bar{m}_1, \bar{m}_2, ..., \bar{m}_{|M|}\}$ and an entity name $S = \{\bar{s}_1, \bar{s}_2, ..., \bar{s}_{|S|}\}$, which were generated by the Representation Layer. We calculate a $|M| \times |S|$-dimensional weight matrix W, whose element $w_{i,j}$ indicates the similarity between the token i of the mention and the token j of the entity name, $w_{ij} = \bar{m}_i^T \bar{s}_j$. Thus, the i^{th} row in W represents the similarity between the i^{th} token in M and each token in S. We apply a softmax function on each row of W to normalize the values, yielding a matrix W'. We can then compute a vector $\tilde{m}_i$ for the i^{th} token of the mention, which is the sum of the vectors of the tokens of S, weighted by their similarity to $\bar{m}_i$:

$$\tilde{m}_i = \sum_{j=1}^{t} w'_{ij} \bar{s}_j \tag{3}$$

This vector "reconstructs" $\bar{m}_i$ by adding up suitable vectors from S, using mainly those vectors of S that are similar to $\bar{m}_i$. If this reconstruction succeeds (i.e., if $\bar{m}_i$ is similar to $\tilde{m}_i$), then S contained tokens which, together, contain the same information as $\bar{m}_i$. To measure this similarity, we could use a simple dot-product. However, this reduces the similarity to a single scalar value, which erases precious element-wise similarities. Therefore, we use the following two comparison functions (Tai, Socher, and Manning 2015; Wang and Jiang 2016):

$$\textit{sub}(\bar{m}_i, \tilde{m}_i) = (\bar{m}_i - \tilde{m}_i) \odot (\bar{m}_i - \tilde{m}_i) \tag{4}$$

$$\textit{mul}(\bar{m}_i, \tilde{m}_i) = \bar{m}_i \odot \tilde{m}_i \tag{5}$$

where the operator $\odot$ means element-wise multiplication. Intuitively, the functions sub and mul represent subtraction and multiplication, respectively. The function *sub* has similarities to the Euclidean distance, while *mul* has similarities to the cosine similarity – while preserving the element-wise information. Finally, we obtain a new representation of each token i of the mention by concatenating $\bar{m}_i, \tilde{m}_i$ and their difference and similarity:

$$\hat{m}_i = [\bar{m}_i, \tilde{m}_i, \textit{sub}(\bar{m}_i, \tilde{m}_i), \textit{mul}(\bar{m}_i, \tilde{m}_i)] \tag{6}$$

By applying the same procedure on the columns of W, we can compute analogously a vector $\tilde{s}_j$ for each token vector s_j of S, and obtain the new representation for the j^{th} token of the entity name as

$$\hat{s}_j = [\bar{s}_j, \tilde{s}_j, \textit{sub}(\bar{s}_j, \tilde{s}_j), \textit{mul}(\bar{s}_j, \tilde{s}_j)] \tag{7}$$

This representation augments the original representation $\bar{s}_j$ of the token by the "reconstructed" token $\tilde{s}_j$, and by information about how similar $\tilde{s}_j$ is to $\bar{s}_j$.

CNN Layer. We now have rich representations for the mention and the entity name, and we apply a one-layer CNN on the mention $[\hat{m}_1, \hat{m}_2, ..., \hat{m}_{|M|}]$ and the entity name $[\hat{s}_1, \hat{s}_2, ..., \hat{s}_{|S|}]$. We adopt the CNN architecture proposed by (Kim 2014) to extract n-gram features of each text:

$$f_M = \textit{CNN}([\hat{m}_1, \hat{m}_2, ..., \hat{m}_M]) \tag{8}$$

$$f_E = \textit{CNN}([\hat{s}_1, \hat{s}_2, ..., \hat{s}_S]) \tag{9}$$

We concatenate these to a single vector $f_{\textit{out}} = [f_M, f_E]$.

Output Layer. We are now ready to compute the final output of our network using a two-layer fully connected neural network:

$$\Phi(M, E) = \textit{sigmoid}(W_2 \ \textit{ReLU}(W_1 \ f_{\textit{out}} + b_1) + b_2) \tag{10}$$

where W_2 and W_1 are learned weight matrices, and b_1 and b_2 are bias values. This constitutes our base model, which relies solely on string similarity. We will now see how we can add add prior, context, and coherence features.

3.4 Extra Features

Mention-Entity Prior. Consider an ambiguous case such as *"You should shower, let water flow over wounds, pat dry with a towel."* appearing in hospital Discharge Instructions. In this context, the disease name *"wounds"* is much more likely to refer to *"surgical wound"* than *"gunshot wound"*. This prior probability is called the *mention-entity prior*. It can be estimated, e.g., by counting in Wikipedia how often a mention is linked to the page of an entity (Hoffart et al. 2011). Unlike DBpedia and YAGO, biomedical knowledge bases generally do not provide links to Wikipedia. Hence, we estimate the mention-entity prior from the training set, as:

$$\textit{prior}(M, E) = \log \textit{count}(M, E) \tag{11}$$

where $\textit{count}(M, E)$ is the frequency with which the mention M is linked to the target entity E in the training dataset. To reduce the effect of overly large values, we apply the logarithm. This prior can be added easily to our model by concatenating it in $f_{\textit{out}}$:

$$f_{\textit{out}} = [f_M, f_E, \textit{prior}(M, E)] \tag{12}$$

Context. The context around a mention can provide clues on which candidate entity to choose. We compute a context score that measures how relevant the keywords of the context are to the candidate entity name. We first represent the sentence containing the mention by pre-trained word embeddings. We then run a Bi-directional LSTM on the sentence to get a new representation for each word. In the same way, we apply a Bi-directional LSTM on the entity name tokens to get the entity name representation cxt_E. To select keywords relevant to the entity while ignoring noise words, we adopt an attention strategy to assign a weight for each token in the sentence. Then we use a weighted sum to represent the sentence as cxt_M. The context score is then computed as the cosine similarity between both representations:

$$\textit{context}(M, E) = \cos(cxt_M, cxt_E) \tag{13}$$

As before, we concatenate this score to the vector f_{out}.

Coherence. Certain entities are more likely to occur together in the same document than others, and we can leverage this disposition to help the entity linking. To capture the co-occurrence of entities, we pre-train entity embeddings in such a way that entities that often co-occur together have a similar distributed representation. We train these embeddings with Word2Vec (Mikolov et al. 2013) on a collection of PubMed abstracts[2]. Since the entities in this corpus are not linked to our KB, we consider every occurrence of an exact entity name as a mention of that entity.

Given a mention M and a candidate entity E, we compute a coherence score to measure how often the candidate entity co-occurs with the other entities in the document. We first select the mentions around M. For each mention, we use the first entity candidate (as given by the candidate selection). This gives us a set of entities $P_M = \{p_1, p_2, ..., p_k\}$, where each element is a pre-trained entity vector. Finally, the coherence score is computed as:

$$\textit{coherence}(M, E) = \frac{1}{k} \sum_{i=1}^{k} \cos(p_i, p_E) \qquad (14)$$

where p_E is the pre-trained vector of the entity candidate E. This score measures how close the candidate entity E is, on average, to the other presumed entities in the document. As before, we concatenate this score to the vector f_{out}. More precisely, we pre-trained separate entity embeddings for the three datasets and used the mean value of all entity embeddings to represent missing entities.

3.5 NIL Problem

The NIL problem occurs when a mention does not correspond to any entity in the KB. We adopt a traditional threshold method, which considers a mention unlinkable if its score is less than a threshold τ. This means that we map a mention to the highest-scoring entity if that score exceeds τ, and to NIL otherwise. The threshold τ is learned from a training set. For datasets that do not contain unlinkable mentions, we set the threshold τ to zero.

3.6 Training

For training, we adopt a triplet ranking loss function to make the score of the positive candidates higher than the score of the negative candidates. The objective function is:

$$\theta^* = \arg\min_{\theta} \sum_{D \in \mathcal{D}} \sum_{M \in D} \sum_{E \in C} \max(0, \gamma + \Phi(M, E^+) - \Phi(M, E^-)) \qquad (15)$$

where θ stands for the parameters of our model. $\mathcal{D}$ is a training set containing a certain number of documents and γ is the parameter of margin. E^+ and E^- represent a positive entity candidate and a negative entity candidate, respectively. Our goal is to find an optimal θ, which makes the score difference between positive and negative entity candidates as large as possible. For this, we need triplets of a mention M, a positive example E^+ and a negative example E^-. The positive example can be obtained from the training set. The negative examples are usually chosen by random sampling from the KB. In our case, we sample the negative example from the candidates that were produced by the candidate generation phase (excluding the correct entity). This choice makes the negative examples very similar to the positive example, and forces the process to learn what distinguishes the positive candidate from the others.

[2]ftp://ftp.ncbi.nlm.nih.gov/pubmed/baseline/

4 Experiments

4.1 Datasets and Metrics.

We evaluate our model on three datasets (shown in Table 1). The **ShARe/CLEF** corpus (Pradhan et al. 2013) comprises 199 medical reports for training and 99 for testing. As Table 1 shows, 28.2% of the mentions in the training set and 32.7% of the mentions in the test set are unlinkable. The reference knowledge base used here is the SNOMED-CT subset of the UMLS 2012AA (Bodenreider 2004). The **NCBI** disease corpus (Doğan, Leaman, and Lu 2014) is a collection of 793 PubMed abstracts partitioned into 693 abstracts for training and development and 100 abstracts for testing. We use the July 6, 2012 version of MEDIC (Davis et al. 2012), which contains 9,664 disease concepts. The TAC 2017 Adverse Reaction Extraction (**ADR**) dataset consists of a training set of 101 labels and a test set of 99 labels. The mentions have been mapped manually to the MedDRA 18.1 KB, which contains 23,668 unique concepts.

Following previous work, we adopt accuracy to compare the performance of different models.

4.2 Experimental Settings

We implemented our model using Keras, and trained our model on a single Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz, using less than 10Gb of memory. Each token is represented by a 200-dimensional word embedding computed on the PubMed and MIMIC-III corpora (Zhang et al. 2019). As for the character embeddings, we use a random matrix initialized as proposed in He et al. (2015), with a dimension of 128. The dimension of the character LSTM is 64, which yields 128-dimensional character feature vectors. In the CNN layer, the number of feature maps is 32, and the filter windows are $[1, 2, 3]$. The dimension of the context LSTM and entity embedding is set to 32 and 50 respectively. We adopt a grid search on a hold-out set from training samples to select the value τ, and and find an optimal for $\tau = 0.75$.

	ShARe/CLEF		NCBI		ADR	
	train	test	train	test	train	test
documents	199	99	692	100	101	99
mentions	5816	5351	5921	964	7038	6343
NIL	1641	1750	0	0	47	18
concepts	88140		9656		23668	
synonyms	42929		59280		0	

Table 1: Dataset Statistics

Model	ShARe/CLEF	NCBI	ADR
DNorm (Leaman, Islamaj Doğan, and Lu 2013)	-	82.20±4.05	-
UWM (Ghiasvand and Kate 2014)	89.50±1.38	-	-
Sieve-based Model (D'Souza and Ng 2015)	90.75±1.31	84.65±3.84	-
TaggerOne (Leaman and Lu 2016)	-	88.80±3.32	-
Learning to Rank (Xu et al. 2017)	-	-	92.05±1.12
CNN-based Ranking (Li et al. 2017)	90.30±1.33	86.10±3.63	-
BERT-based Ranking (Ji, Wei, and Xu 2020)	**91.06±1.29**	89.06±3.32	**93.22±1.04**
Our Base Model	90.10±1.35	89.07±3.32	92.89±1.06
Our Base Model + Extra Features	90.43±1.33	**89.59±3.22**	93.00±1.06

Table 2: Performance of different models. Results in gray are not statistically different from the top result.

During the training phase, we select at most 20 entity candidates per mention, and the parameter of the triplet rank loss is 0.1. For the optimization, we use Adam with a learning rate of 0.0005 and a batch size of 64. To avoid overfitting, we adopt a dropout strategy with a dropout rate of 0.1.

4.3 Competitors

We compare our model to the following competitors: **DNorm** (Leaman, Islamaj Doğan, and Lu 2013); **UWM** (Ghiasvand and Kate 2014); **Sieve-based Model** (D'Souza and Ng 2015); **TaggerOne** (Leaman and Lu 2016); a model based on **Learning to Rank** (Xu et al. 2017); **CNN-based Ranking** (Li et al. 2017); and **BERT-based Ranking** (Ji, Wei, and Xu 2020).

5 Results

5.1 Overall Performance

During the candidate generation, we generate 20 candidates for each mention. The recall of correct entities on the ShARe/CLEF, NCBI, and ADR test datasets is 97.79%, 94.27%, and 96.66% respectively. We thus conclude that our candidate generation does not eliminate too many correct candidates. Table 2 shows the performance of our model and the baselines. Besides accuracy, we also compute a binomial confidence interval for each model (at a confidence level of 0.05), based on the total number of mentions and the number of correctly mapped mentions. The best results are shown in bold text, and all performances that are within the error margin of the best-performing model are shown in gray. We first observe that, for each dataset, several methods perform within the margin of the best-performing model. However, only two models are consistently within the margin across all datasets: BERT and our method. Adding extra features (prior, context, coherence) to our base model yields a small increase on the three datasets. However, overall, even our base model achieves a performance that is statistically indistinguishable from the state of the art.

5.2 Ablation Study

To understand the effect of each component of our model, we measured the performance of our model when individual components are removed or added. The results of this ablation study on all three datasets are shown in Table 3. The gray row is the accuracy of our base model. The removal of the components of the base model is shown above the gray line; the addition of extra features (Section 3.4) below. If we remove the Alignment Layer (underlined), the accuracy drops the most, with up to 4.06 percentage points. This indicates that the alignment layer can effectively capture the similarity of the corresponding parts of mentions and entity names. The CNN Layer extracts the key components of the names, and removing this part causes a drop of up to 1.87 percentage points. The character-level feature captures morphological variations, and removing it results in a decrease of up to 1.21 percentage points. Therefore, we conclude that all components of our base model are necessary.

Model	ShARe/CLEF	NCBI	ADR
- Character Feature	-1.21	-0.31	-0.30
- Alignment Layer	-3.80	-4.06	-3.17
- CNN Layer	-1.87	-0.93	-0.35
Our Base Method	90.10	89.07	92.89
+ Mention-Entity Prior	+0.33	+0.04	+0.03
+ Context	-0.09	+0.21	-0.24
+ Coherence	-0.02	+0.27	+0.11

Table 3: Ablation study

Let us now turn to the effect of the extra features of our model. The Mention-Entity Prior can bring a small improvement, because it helps with ambiguous mentions, which occupy only a small portion of the dataset. The context feature, likewise, can achieve a small increase on the NCBI dataset. On the other datasets, however, the feature has a negative impact. We believe that this is because the documents in the NCBI datasets are PubMed abstracts, which have more relevant and informative contexts. The documents in the ShARe/CLEF and ADR datasets, in contrast, are more like semi-structured text with a lot of tabular data. Thus, the context around a mention in these documents is less helpful. The coherence feature brings only slight improvements. This could be because our method of estimating co-occurrence is rather coarse-grained, and the naive string matching we use may generate errors and omissions. In conclusion, the extra features do bring a small improvement, and they are thus an interesting direction of future work. However, our simple base model is fully sufficient to achieve state-of-the-art performance already.

Model	Original ADR	10%	30%	50%	70%	90%
+ Ordering Change	92.89	92.46	92.44	92.23	92.57	92.31
+ Typo	92.89	92.29	91.87	91.64	91.67	91.39

Table 4: Performance in the face of typos: Simulated ADR Datasets

Model	Parameters	ShARe/CLEF		NCBI		ADR		Avg	Speedup
		CPU	GPU	CPU	GPU	CPU	GPU		
BERT (large)	340M	2230s	1551s	353s	285s	2736s	1968s	1521s	12.3x
BERT (base)	110M	1847s	446s	443s	83s	1666s	605s	848s	6.4x
TinyBERT$_6$	67M	1618s	255s	344s	42s	2192s	322s	796s	6.0x
MobileBERT (base)	25.3M	1202s	330s	322s	58s	1562s	419s	649s	4.7x
ALBERT (base)	12M	836s	**129s**	101s	24s	1192s	170s	409s	2.6x
Our Base Model	4.6M	**181s**	131s	**38s**	**22s**	**196s**	**116s**	**114s**	-

Table 5: Number of model parameters and observed inference time

5.3 Performance in the face of typos

To reveal how our base model works, we further evaluate it on simulated ADR datasets. We generate two simulated datasets by randomly adding typos and changing word orderings of mention names. As described in Table 4, as we gradually add typos, the accuracy does not drop too much, and adding 90% of typos only results in a 1.5 percent drop. This shows our model can deal well with morphological variations of biomedical names. Besides, ordering changes almost have no effect on our base model, which means it can capture correspondences between mention and entity names.

5.4 Parameters and Inference Time

To measure the simplicity of our base model, we analyze two dimensions: the number of model parameters and the practical inference time. In Table 5, we compare our model with BERT models, including three popular lightweight models: ALBERT(Lan et al. 2019), TinyBERT(Jiao et al. 2019), and MobileBert(Sun et al. 2020). Although ALBERT's size is close to our model, its performance is still 2.2 percentage points lower than the BERT$_{BASE}$ model on average.

The second column in the table shows the number of parameters of different models. Our model uses an average of only 4.6M parameters across the three data sets, which is 1.6x to 72.9x smaller than the other models. The third column to the tenth column show the practical inference time of the models on the CPU and GPU. The CPU is described in Section 4.2, and the GPU we used is a single NVIDIA Tesla V100 (32G). Our model is consistently the fastest across all three datasets, both for CPU and GPU (except in the fourth column). On average, our model is 6.4x faster than other BERT models, and our model is much lighter on the CPU.

5.5 Model Performance as Data grows

In this section, we study how our model performs with an increasing amount of training samples, by subsampling the datasets. As shown in Figure 2, the performance of our base model keeps growing when we gradually increase the number of training samples. When using 50% of the training samples, the accuracies of ShARe/CLEF, NCBI, and ADR dataset are already 0.8342, 0.8747, and 0.9106, respectively. More data leads to better performance, and thus our model is not limited by its expressivity, even though it is very simple.

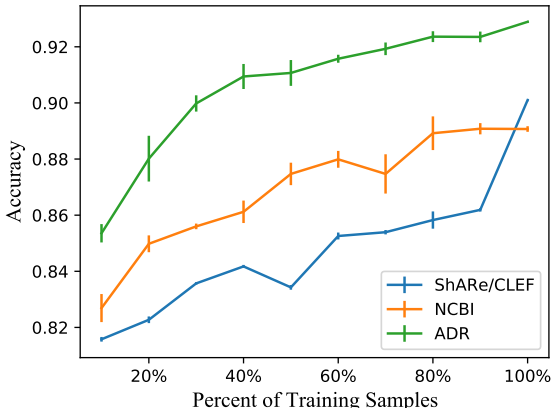

Figure 2: Model efficiency on a small amount of data.

6 Conclusion

In this paper, we propose a simple and lightweight neural model for biomedical entity linking. Our experimental results on three standard evaluation benchmarks show that the model is very effective, and achieves a performance that is statistically indistinguishable from the state of the art. BERT-based models, e.g., have 23 times more parameters and require 6.4 times more computing time for inference. Future work to improve the architecture can explore *1)* automatically assigning a weight for each word in the mentions and entity names to capture the importance of each word, depending, e.g., on its grammatical role; *2)* Graph Convolutional Networks (GCNs) (Kipf and Welling 2016; Wu et al. 2020) to capture graph structure across mentions and improve our notion of entity coherence.

Acknowledgments. This project was partially funded by the DirtyData project (ANR-17-CE23-0018-01).

References

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, 722–735. Springer.

Bodenreider, O. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research* 32(suppl_1): D267–D270.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038* .

Davis, A. P.; Wiegers, T. C.; Rosenstein, M. C.; and Mattingly, C. J. 2012. MEDIC: a practical disease vocabulary used at the Comparative Toxicogenomics Database. *Database* 2012.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Doğan, R. I.; Leaman, R.; and Lu, Z. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics* 47: 1–10.

Dogan, R. I.; and Lu, Z. 2012. An inference method for disease name normalization. In *2012 AAAI Fall Symposium Series*.

D'Souza, J.; and Ng, V. 2015. Sieve-based entity linking for the biomedical domain. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 297–302.

Ghiasvand, O.; and Kate, R. J. 2014. R.: UWM: Disorder mention extraction from clinical text using CRFs and normalization using learned edit distance patterns. In *In: Proc. SemEval 2014*. Citeseer.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6645–6649. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Hoffart, J.; Yosef, M. A.; Bordino, I.; Fürstenau, H.; Pinkal, M.; Spaniol, M.; Taneva, B.; Thater, S.; and Weikum, G. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 782–792.

Huang, K.; Altosaar, J.; and Ranganath, R. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342* .

Ji, Z.; Wei, Q.; and Xu, H. 2020. Bert-based ranking for biomedical entity normalization. *AMIA Summits on Translational Science Proceedings* 2020: 269.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* .

Kang, N.; Singh, B.; Afzal, Z.; van Mulligen, E. M.; and Kors, J. A. 2013. Using rule-based natural language processing to improve disease normalization in biomedical text. *Journal of the American Medical Informatics Association* 20(5): 876–881.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* .

Leaman, R.; Islamaj Doğan, R.; and Lu, Z. 2013. DNorm: disease name normalization with pairwise learning to rank. *Bioinformatics* 29(22): 2909–2917.

Leaman, R.; and Lu, Z. 2016. TaggerOne: joint named entity recognition and normalization with semi-Markov Models. *Bioinformatics* 32(18): 2839–2846.

Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C. H.; and Kang, J. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36(4): 1234–1240.

Li, H.; Chen, Q.; Tang, B.; Wang, X.; Xu, H.; Wang, B.; and Huang, D. 2017. CNN-based ranking for biomedical entity normalization. *BMC bioinformatics* 18(11): 79–86.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Pradhan, S.; Elhadad, N.; South, B. R.; Martinez, D.; Christensen, L. M.; Vogel, A.; Suominen, H.; Chapman, W. W.; and Savova, G. K. 2013. Task 1: ShARe/CLEF eHealth Evaluation Lab 2013. In *CLEF (Working Notes)*, 212–31.

Sohn, S.; Comeau, D. C.; Kim, W.; and Wilbur, W. J. 2008. Abbreviation definition identification based on automatic precision estimates. *BMC bioinformatics* 9(1): 402.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, 697–706.

Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; and Zhou, D. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984* .

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Wang, S.; and Jiang, J. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747* .

Wright, D. 2019. *NormCo: Deep disease normalization for biomedical knowledge base construction*. Ph.D. thesis, UC San Diego.

Wu, J.; Zhang, R.; Mao, Y.; Guo, H.; Soflaei, M.; and Huai, J. 2020. Dynamic Graph Convolutional Networks for Entity Linking. In *Proceedings of The Web Conference 2020*, 1149–1159.

Xu, J.; Lee, H.-J.; Ji, Z.; Wang, J.; Wei, Q.; and Xu, H. 2017. UTH_CCB System for Adverse Drug Reaction Extraction from Drug Labels at TAC-ADR 2017. In *TAC*.

Zhang, Y.; Chen, Q.; Yang, Z.; Lin, H.; and Lu, Z. 2019. BioWordVec, improving biomedical word embeddings with subword information and MeSH. *Scientific data* 6(1): 1–9.