NSEEN: Neural Semantic Embedding for Entity Normalization

Shobeir Fakhraei(✉), Joel Mathew, and José Luis Ambite

Information Sciences Institute, University of Southern California, Los Angeles, USA
{shobeir,joel,ambite}@isi.edu

Abstract. Much of human knowledge is encoded in text, available in scientific publications, books, and the web. Given the rapid growth of these resources, we need automated methods to extract such knowledge into machine-processable structures, such as knowledge graphs. An important task in this process is *entity normalization*, which consists of mapping noisy entity mentions in text to canonical entities in well-known *reference sets*. However, entity normalization is a challenging problem; there often are many textual forms for a canonical entity that may not be captured in the reference set, and entities mentioned in text may include many syntactic variations, or errors. The problem is particularly acute in scientific domains, such as biology. To address this problem, we have developed a general, scalable solution based on a deep Siamese neural network model to embed the *semantic* information about the entities, as well as their *syntactic* variations. We use these embeddings for fast mapping of new entities to large reference sets, and empirically show the effectiveness of our framework in challenging bio-entity normalization datasets.

Keywords: Semantic embedding · Deep learning · Siamese networks · Entity grounding · Entity normalization · Entity resolution · Entity disambiguation · Entity matching · Data integration · Similarity search · Similarity learning

1 Introduction

Digital publishing has accelerated the rate of textual content generation to beyond human consumption capabilities. Taking the scientific literature as an example, Google Scholar has indexed about four and a half million articles and books in 2017 in a 50% increase from the previous year. Automatically organizing this information into a proper knowledge representation is an important way to make this information accessible. This process includes identification of entities in the text, often referred to as *Names Entity Recognition (NER)* [25,38], and mapping of the identified entities to existing reference sets, called *Entity Normalization*, or *Grounding*. In this paper we propose a text embedding solution for entity normalization to a reference set.

U. Brefeld et al. (Eds.): ECML PKDD 2019, LNAI 11907, pp. 665–680, 2020.
https://doi.org/10.1007/978-3-030-46147-8_40

Entity normalization to a reference set is a challenging problem. Even though in some cases normalization can be as simple as a database look-up, often there is no exact match between the recognized entity in the text and the reference entity set. There are two main sources for this variation. The first is *syntactic variations*, where the identified entity contains relatively small character differences with the canonical form present in the reference set, such as different capitalization, reordering of words, typos, or errors introduced in the NER process (e.g., 'FOXP2' and 'FOX-P2'). The second and more challenging problem, which we call *semantic variations*, is when the identified entity does not exist in the reference set, even when considering significant syntactic variations, but a human reader can recognize the non-standard entity name. For example, entities often have multiple canonical names in the reference set and the identified entity name is a combination of parts of different canonical names (e.g., 'P70 S6KA' and '52 kDa ribosomal protein S6 kinase').

A further challenge is how to perform normalization at scale. Exhaustive pairwise comparison of the identified entity to the reference entities grows quadratically and is unfeasible for large datasets. *Blocking* [31] techniques speed up the process by selecting small subsets of entities for pairwise comparisons. Unfortunately, blocking methods applied directly to the textual representation of the entity names are often limited to simple techniques that can only address syntactic variations of the entity names. So, traditional blocking may eliminate matches that are semantically relevant but syntactically different.

To address these issues, we develop a text embedding solution for entity normalization. Our contributions include: (1) A general, scalable deep neural-based model to embed entity information in a numeric vector space that captures both *syntactic* and *semantic variations*. (2) An approach to incorporate syntactic variations of entity names into the embeddings based on domain knowledge by extending the use of contrastive loss function with soft labels. (3) A method for dynamic hard negative mining to refine the embedding for improved performance. (4) Using an approximate *k-nearest neighbors* algorithm over the embeddings to provide a scalable solution without the need for traditional blocking.

2 Related Work

Data Normalization, linking entities to their canonical forms, is one of the most fundamental tasks in information retrieval and automatic knowledge extraction [9]. Many related tasks share components with entity normalization, but also have subtle differences. Record linkage [21], aims to find records from different tables corresponding to the same entity. Records often contain multiple fields and one of the challenges in this task is reasoning on different fields, and their combinations. Deduplication [13] is similar to record linkage, but focuses on the records of the same table, so it does not have to consider the heterogeneity of fields across different tables. Entity resolution [14], is a more general term that deals with findings entity mentions that refer to the same entity and often inferring a canonical form from them.

A critical feature in our setting is the presence of a canonical *reference set*, so that we ask "which canonical entity a mention is mapped to?" in contrast to "which records are the same?" for settings were the canonical entity is latent. Reference sets are specially important in bio-medical domains [23]. Unlike record linkage, we do not have multiple fields and only reason on a single string.

Feature-engineered string similarities [7] form the core of most traditional entity resolution methods. In contrast, Our approach learns a *similarity metric* for entity normalization based on syntactic and semantic information. We compute these similarities via embedding the entity mentions into a vector space. Text embeddings, such as word2vec [27], GloVe [32], or more recently ELMo [33], and BERT [12] have been very successful in language processing and understanding applications, in great measure because they have been computed over very large corpora. However, these methods are not task specific and provide general embeddings based on the text context. Our approach is based on computing direct similarities rather than analyzing the surrounding text. Hence, for Entity Normalization, we use a deep Siamese neural network that has been shown to be effective in learning similarities in text [30] and images [37]. Both of these approaches define a contrastive loss functions [15] to learn similarities. Recently, Ebraheem et al. [17] and Mudgal et al. [28] proposed deep neural network methods for record linkage (with multiple fields) in a database. A major focus of their work was on combining data in different fields. Our setting differs since we operate on entity name strings, and match them to canonical references.

To avoid exhaustive pairwise computation of similarities between entities often blocking [26] or indexing [10] techniques are used to reduce the search space. These methods are often based on approximate string matching. The most effective methods in this area is based on hashing the string with the main purpose of blocking the entities as a pre-processing step, followed by the matching part that is performed after blocking. In our method, we combine both steps by mapping the entity mentions to a numerical space to capture similarities. The blocking in our case conceptually follows the matching process via applying approximate nearest neighbors approaches on our semantic embedding space.

In the biomedical domain, Kang et al. [19] propose a rule-based method and Leaman et al. [22] propose a learning-to-rank-based approach for disease normalization. Leaman and Lu [23] further perform joint name entity recognition and normalization. We provide an embedding-based approach for entity normalization. We perform our experimental validation on two biomedical datasets of protein and chemical entities.

3 Approach

Problem Definition: Given a query entity mention (n_q), and a reference set of entities $\mathcal{R} \equiv \{e_1, \ldots, e_m\}$, where each entity $e_i \equiv <\lambda_i, \{n_i^1, \ldots, n_i^k\}>$ is identified via an ID (λ_i) and an associated set of names (n_i^k) that refer to the entity, our goal is to return the ID (λ_q) of the corresponding entity in our reference set $\mathcal{R}$. The exact textual name of the query entity may not exist in the reference set.

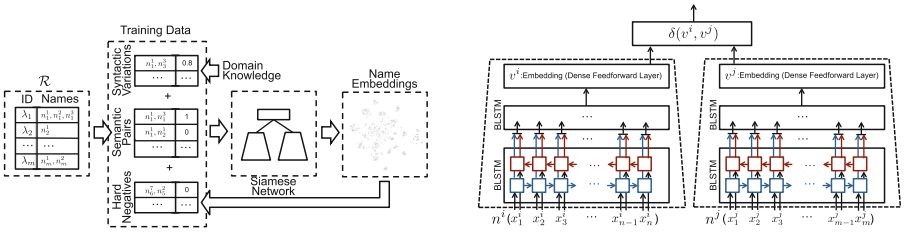


Fig. 1. Learning embedding function based on semantics in reference set and syntactic variations defined by domain knowledge and hard negative mining.

We map this normalization task to an approximate nearest-neighbors search in a n-dimensional space where each name n_l^m in the reference set is encoded into a numerical vector representation v_l^m. Our objective in this embedding space is that names of the same entity (even syntactically very different) be closer to each other compared to names of other entities. That is, $n_l^m \rightarrow v_l^m$ such that $\delta(v_l^m, v_l^p) < \delta(v_l^m, v_o^*)$, where e_l and e_o are entities ($e_l \neq e_o$), n_*^* their corresponding names, v_*^* embedding vectors of these names, and δ is a distance function.

We use a Siamese neural network architecture to embed the semantic information about the entities as well as their syntactic similarities. We further refine the similarities via dynamic hard negative sampling and incorporating domain knowledge about the entities using additional generated training data. We then encode and store the embeddings in a numeric representation that enables fast retrieval of the results without the need for traditional character-based blocking. Our approach consist of three steps:

Similarity Learning. We first learn an embedding function ($\mathcal{M} : n \rightarrow v$) that maps the entity names to a numeric vector space where names of the same entities are close to each other.

Embedding and Hashing. Then, we embed all the names in the reference set $\mathcal{R}$ to the numerical vector space and hash and store the reference set embeddings for fast retrieval.

Retrieval. Finally, we embed the query name (i.e., $n_q \rightarrow v_q$) using the learned model $\mathcal{M}$ and find the closest samples to it in the embedding space to retrieve the corresponding ID (λ_q) of the query name in the reference set.
The following sections describe each step in detail.

3.1 Similarity Learning

We first learn a function ($\mathcal{M}$) that maps the textual representation of entity names (n) to a numerical vector representation (v) that preserves the proximity of names that belong to the same entity, using a Siamese recurrent neural network model. Figure 1(a) shows the overall approach and Algorithm 1 describes the similarity learning process.

Algorithm 1. NSEEN: Similarity Learning

procedure TRAINSIM($\mathcal{R}, P_d$)
  Input: $\mathcal{R}$ reference set
  Input: P_d pairs based on knowledge of syntactic variation in the domain
  Generate pairs based on reference set $\mathcal{R}$ and add them to training data $\mathcal{D}$
  Add P_d pairs to the training data $\mathcal{D}$
  for k times **do**
    Train the model $\mathcal{M}$ (Siamese network) on $\mathcal{D}$
    Embed all the names in $\mathcal{R}$: $n \rightarrow v$
    for all v_l^i **do** ▷ Hard negative mining
      find the k closest v_k^j to v_l^i
      if $k \neq l$ **then**
        add $< n_k^j, n_l^i, 0 >$ to training data $\mathcal{D}$
  return $\mathcal{M}$ ▷ The trained embedding function

Siamese Recurrent Neural Network. The Siamese neural network architecture of two towers with shared weights and a distance function at the last layer has been effective in learning similarities in domains such as text [30] and images [37]. Figure 1(b) depicts an overview of the network used in our framework.

We feed pairs of names and a score indicating the similarity of the pairs (i.e., $<n^i, n^j, y>$) to the Siamese network. As shown in Fig. 1(b), n^i and n^j are entity names represented as a sequences of characters $<x_1^i, \ldots, x_n^i>$ and $<x_1^j, \ldots, x_m^j>$, and $y \in [0, 1]$ represents the similarity between the names. To read the character sequence of the names, we feed the character embedding to four layers of Bidirectional-LSTM, followed by a single densely connected feed-forward layer, which generate the embeddings v.

Contrastive Loss Function. While we can use several distance functions (δ) to compare the learned vectors of the names, we use cosine distance between the embeddings v_i and v_j, due to its better performance in higher dimensional spaces. We then define a contrastive loss [15] based on the distance function δ to train the model, as shown in Eq. 1. The intuition behind this loss function is to pull the similar pairs closer to each other, and push the dissimilar pairs up to a margin m apart ($m = 1$ in our experiments).

$$\mathcal{L} = \frac{1}{2} y \delta(v_i, v_j)^2 + \frac{1}{2}(1 - y) \max(0, m - \delta(v_i, v_j))^2 \quad (1)$$

The contrastive loss has been originally proposed for binary labels where we either fully pull two points towards each other or push them apart. In this paper, we propose to extend this loss via using soft real-valued labels when we introduce syntactic variations of the names described in Sect. 3.1 to indicate uncertainties about the similarities of two vectors. For the margin of 1 (i.e., $m = 1$), the distance that minimizes the loss function $\mathcal{L}$ for the real-valued label y is:[1]

[1] For brevity of notation we denote $\delta(v_i, v_j)$ with δ_v.

$$\frac{\partial \mathcal{L}}{\partial \delta_v} = y\delta_v - (1-y)(1-\delta_v)$$
$$\arg\min_{\delta_v} \mathcal{L} = \{\delta_v \mid y + \delta_v - 1 = 0\} = 1 - y \tag{2}$$

For example, in our setting the optimal distance between the embeddings of two names with 0.7 similarity (i.e., $y = 0.7$) is 0.3. Figure 2 depicts the changes in loss when altering the distance corresponding to different y values, and the points that minimize the loss (i.e., $\arg\min_{\delta_v} \mathcal{L}$) are marked on each line.

Pair Selection and Generation. In order to train the model we need labeled pairs of names ($<n^i, n^j, y>$). We generate three sets of pairs using different approaches: (1) the *initial set* based on the names in the reference set, (2) the *syntactic variations set* based on domain knowledge, and (3) the *hard negative set*. The initial and the hard negative pairs capture the *semantic* relationships between names in the reference set, and the syntactic variations capture the *syntactic* noise that may be present in referring to these names in reality.

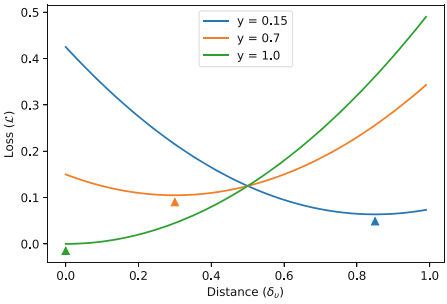


Fig. 2. Contrastive loss ($\mathcal{L}$) based on distance values (δ_v) for different real-value labels y. (Best viewed in color) (Color figure online)

Initial Semantic Set. We generate an initial training set of similar and dissimilar pairs based the entities in the reference set $\mathcal{R}$. We generate positive pairs by the cross product of all the names that belong to the same entity, and initialize the negative set of dissimilar pairs by randomly sampling names that belong to different entities. Formally:

$$P_+ = \{< n^i, n^j, 1 > \mid (\forall n_l^i, n_l^j \in e_l) \wedge (\forall e_l \in \mathcal{R})\}$$
$$P_- = \{< n^i, n^j, 0 > \mid (n_l^i, n_m^j \in e_l, e_m) \wedge (e_l, e_m \in \mathcal{R}) \wedge (e_l \neq e_m)\}$$

Syntactic Variations and Entity Families. In order to train the model with the syntactic variations that could be introduced in the real-world textual representation of the names, we add pairs of names to the training set and label them with their real-value string similarities. The argument behind using real-valued labels is provided in Eq. 2, with the intuition that using a label of 0 will completely repel two vectors and using a label of 1 will bring two vectors as close as possible, but using a label between 0 and 1 will aim to keep the two vectors somewhere inside the margin.

We use Trigram-Jaccard, Levenshtein Edit Distance, and Jaro–Winkler to compute string similarity scores [11] between the pairs of names and include

sets of pairs with labels based on each similarity score in the training set. The intuition is that the model will learn a combination of all these string similarity measures. To select the name pairs to include in this process, we consider two sets of variations based on the *same name*, and *different names*.

Same name variations are the noise that can be introduced to an extracted name in real-world settings. To capture the most common forms of noise occurring on the same name, we make the following three modifications based on our observation of the most frequent variations in the query names:

- Removing the spaces, e.g., <FOX P2, FOXP2, y>
- Removing all but alphanumerical characters, e.g., <FOX-P2, FOXP2, y>
- Converting to upper and lower cases, e.g., <Ras, RAS, y>, <Ras, ras, y>

Different name variations introduce a higher level of similarity concept to the model. We make the second set of pairs by selecting the names of entities that are *somehow related* and computing their string similarities. For example, in our experiments with proteins we select two entities that belong to the same protein family and generate pairs of names consisting of one name from each. The labels are assigned to these pairs based on their string similarities. This set of pairs not only introduces more diverse variations of textual string similarities, it also captures a *higher-level relationship* by bringing the embeddings of the names that belong to a group closer to each other. Encoding such hierarchical relations in the entity representations has been effective in various domains [8].

Hard Negative Mining. Given the large space of possible negative name pairs (i.e., the cross product of the names of different entities) we can only sample a subset to train our model. As stated earlier we start with an initial random negative sample set for our training. However, these random samples may often be trivial choices for the model and after a few epochs may not contain enough useful signal. The use of contrastive loss makes this issue more problematic as the probability of the distance between randomly selected negative samples being less than the margin (m) is low. Sampling techniques, often called *hard-negative mining*, have been introduces in domains such as knowledge graph construction [20] and computer vision [36] to deal with similar issues.

The idea behind hard negative mining is finding negative examples that are most informative for the model. These examples are the ones closest to the decision boundary and the model will most likely assign a wrong label to them. As shown in Fig. 1a and Algorithm 1, we find the hard negatives by first embedding all the names in the reference set $\mathcal{R}$ using the latest learned model $\mathcal{M}$. We then find the closest names to each name in the embedding space using an approximate k-nearest neighbors algorithm for fast iterations. We then add the name pairs found using this process that do not belong to the same entity with a 0 label to our training set and retrain the model $\mathcal{M}$. We repeat this process multiple times to refine the model with several sets of hard negative samples.

3.2 Reference Set Embedding and Storage

The model $\mathcal{M}$ that we trained in the previous step is basically a function that maps a name string to a numerical vector. Since both towers of the Siamese network share all their weights, the final embedding is independent of the tower the original string is provided to as input. Considering the goal of our framework, which is to perform entity normalization of query names (n_q) to the entities in the reference set $\mathcal{R}$, we embed all the names in the reference set using the final trained model $\mathcal{M}$, and store the embeddings for comparison with future queries.

Our task becomes assigning an entity in our reference set to the query name n_q by finding the closest entity to it in the embedding space. This assignment is basically a nearest neighbor search in the embedding space. The most naive solution to this search would entail a practically infeasible task of exhaustive pairwise comparisons of query embedding with all embeddings in a potentially large reference set. Moreover, since we iteratively repeat the nearest neighbor look-up in our training process for hard-negative mining, we need a faster way to retrieve the results.

This challenge is prevalent in many research and industry applications of machine learning such as recommender systems, computer vision, and in general any similarity-based search, and has resulted in development of several fast *approximate nearest neighbors* approaches [34,35]. We speed-up our nearest neighbors retrieval process by transforming and storing our reference set embeddings in an approximate nearest neighbors data structure. Algorithm 2 describes the overall process of this stage.

Algorithm 2. Embedding $\mathcal{R}$

procedure EMBED($\mathcal{R}, \mathcal{M}$)
  for all $n_i \in \mathcal{R}$ **do**
    $n_i \xrightarrow{\mathcal{M}} v_i$
  for all v_i **do**
    Hash v_i and store in $\mathcal{H}_{v_i}$
  return $\mathcal{H}_v$ ▷ Hashed embeddings

Algorithm 3. Retrieval

procedure RETRIEVE($\mathcal{H}_v$, $\mathcal{M}$, n_q)
  Embed the query name: $n_q \xrightarrow{\mathcal{M}} v_q$
  Find the closest v_k^j to v_q using approximate nearest neighbor search (Annoy) on $\mathcal{H}_v$
  return λ_k as the ID (i.e., λ_q)

We leverage a highly optimized solution that is extensively used in applied settings, such as Spotify, to deal with large scale approximate nearest neighbor search, called *Annoy (Approximate Nearest Neighbors Oh Yeah!)* [2]. Annoy, uses a combination of random projections and a tree structure where intermediate nodes in the tree contain random hyper-planes dividing the search space. It supports several distance functions including Hamming and cosine distances based on the work of Bachrach et al. [5].

Since we have already transformed the textual representation of an entity name to a numerical vector space, and the entity look-up to a nearest neighbor search problem, we can always use competing approximate nearest neighbors search methods [29], and the new state-of-the-art approaches that will be discovered in the future. Furthermore, using such scalable data structures for our embeddings at this stage preserves semantic similarities learned by our model,

in contrast to traditional blocking approaches applied as a pre-processing step that could break the semantic relationship in favor of textual similarities.

3.3 Retrieval

During the retrieval step, depicted in Algorithm 3 we first compute an embedding for the query name based on the same model $\mathcal{M}$ that we used to embed the reference set. We then perform an approximate nearest neighbor search in the embedding space for the query name, and return the ID of retrieved neighbor as the most probable entity ID for the query name. Note that in our setup we do not need to perform a separate direct look up for the query names that *exactly* match one of canonical names in the reference set. If the query name is one of the canonical names in the reference set, it will have exactly the same embedding and zero distance with one of the reference set names.

4 Experimental Validation

We conduct two set of experiments mapping query names to their canonical names to empirically validate the effectiveness of our framework. The two references sets are *UniProt* for proteins and *ChEBI* for chemical entities, and the query set is from PubMed extracts provided by the BioCreative initiative [4], as detailed in the following sections.

4.1 Reference Sets

The reference sets we use in our experiments are publicly available on the internet, and are the authority of canonical entity representations in their domains.

UniProt. The Universal Protein Resource (UniProt) is a large database of protein sequences and associated annotations [3]. For our experiments, we use the different names associated with each human protein in the UniProt dataset and their corresponding IDs. Hence, the task here is mapping a human protein name to a canonical UniProt ID.

ChEBI. We used the chemical entity names indexed in the Chemical Entities of Biological Interest (ChEBI) ontology. ChEBI is a dataset of molecular entities focused on small chemical compounds, including any constitutionally or isotopically distinct atom, molecule, ion, ion pair, radical, radical ion, complex, conformer, identifiable as a separately distinguishable entity [16]. The task here is mapping a small molecule name to a canoncal ChEBI ID.

Table 1 depicts the total number of entities (e_i) and their corresponding ID–name pairs ($<\lambda_i, n_i^j>$) in the reference sets, showing UniProt having less number of entities, but more names per entity comparing to ChEBI. Moreover, Fig. 3 depicts the histogram that shows the distribution of the number of names per each entity in the reference sets. Note that there are no entities in the UniProt reference set with only one name, but there are many proteins with several names. In contrast, the ChEBI dataset contains many entities with only one name.

Table 1. Statistics of the entities in the reference sets

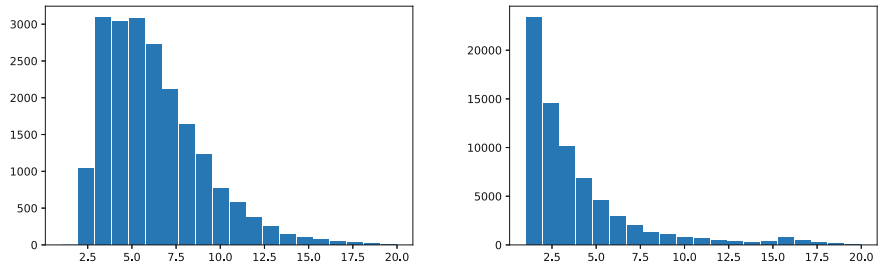
Datasets	Entities	<entity, name> pairs
UniProt (Human)	20,375	123,590
ChEBI	72,241	277,210

4.2 Query Set

We use the datasets provided by the BioCreative VI Interactive Bio-ID Assignment Track [4] as our query data. These datasets provide several types of biomedical entity annotations generated by SourceData curators that map published article texts to their corresponding database IDs. The main interesting point about the BioCreative corpus for entity normalization is that the extracted entity names come from published scientific articles, and contain the entity-name variations and deviations forms that are present in the real world.

The Bio-ID datasets include a separate *train* and *test* sets. We use both of these datasets as query sets with gold standard labels to evaluate our method. The training set (we name it BioC1) consists of 13,573 annotated figure panel captions corresponding to 3,658 figures from 570 full length articles from 22 journals, for a total of 102,717 annotations. The test data set (we name it BioC2) consisted of 4,310 annotated figure panel captions from 1,154 figures taken from 196 full length journal articles, with 30,286 annotations in total [4].

Table 2 shows the number of UniProt and ChEBI entities in the annotated corpus. In our experiments we keep the original training (BioC1) and test (BioC2) splits of the data for reproducibility and ease of future comparisons, but we should note that for our purposes both BioC1 and BioC2 are just a source of correct normalizations with gold standards, and test sets in our experiments. Our algorithm is not trained on any of these datasets.



(a) UniProt (b) ChEBI

Fig. 3. Distribution of names per entity in the reference datasets.

Table 2. Statistics of the annotations in the BioCreative VI Bio-ID corpus

Dataset	UniProt		ChEBI	
	Mentions	Entities	Mentions	Entities
BioC1	30,211	2,833	9,869	786
BioC2	1,592	1,321	829	543

4.3 Baselines

USC–ISI. As a representative of traditional record linkage techniques, we use the current production system for Named Entity Grounding at USC Information Science Institute, developed for the DARPA Big Mechanism program, as one of the baselines. The system is an optimized solution that employs a tuned combination of several string similarities including Jaccard, Levenshtein, and JaroWinkler distances with a prefix-based blocking system. It also includes a post re-ranking of the results based on the domain knowledge, such as the curation level of the entity (e.g., if the protein entry in UniProt has been reviewed by a human or not), the matching between the ID components and the query name, and popularity of the entities in each domain. This system provides entity grounding for several bio-medical entities including Proteins and Chemicals, and is publicly available at [1]. The system can produce results based on the FRIL [18] record linkage program and Apache Lucene [6], and we use the overall best results of both settings as the baseline for our experiments. We chose this baseline as a representative of the traditional entity normalization methods that provides competitive results based on an ensemble of such models.

BioBERT. To compare our method with a representative of text embedding approaches, we used the embedding generated by the recently released BioBERT [24] (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining) model which extends the BERT [12] approach. BioBERT is a domain specific language representation model pre-trained on large-scale biomedical corpora that can effectively capture knowledge from a large amount of biomedical texts with minimal task-specific architecture modifications. BioBERT outperforms traditional models in biomedical named entity recognition, biomedical relation extraction, and biomedical question answering. We used the BioBERT framework with pre-trained weights released by the original authors of the paper, in a similar process to our approach; we first embed all the entity names of the reference set and then find the closest embedding to the query name in that embedding space.

DeepMatcher. Mudgal et al. [28] recently studied the application of deep learning architectures on entity matching in a general setting where the task is matching tuples (potentially having multiple fields) in different tables. DeepMatcher outperforms traditional entity matching frameworks in textual and noisy settings. We use DeepMatcher as a representative baseline for deep learning methods specific to entity normalization.

Table 3. Hits@k on BioCreative train dataset (BioC1) and test dataset (BioC2) datasets mapped to Uniprot and ChEBI reference sets.

Reference($\mathcal{R}$)	Dataset	Method	H@1	H@3	H@5	H@10
UniProt	BioC1	DeepMatcher	0.697	0.728	0.739	0.744
		BioBERT	0.729	0.761	0.779	0.808
		USC–ISI	0.814	0.864	0.875	0.885
		NSEEN	**0.833**	**0.869**	**0.886**	**0.894**
	BioC2	DeepMatcher	0.767	0.792	0.803	0.814
		BioBERT	0.801	0.827	0.827	0.840
		USC–ISI	0.841	0.888	0.904	0.919
		NSEEN	**0.861**	0.888	0.904	**0.930**
ChEBI	BioC1	DeepMatcher	0.288	0.363	0.397	0.419
		BioBERT	0.360	0.473	0.499	0.524
		USC–ISI	0.418	0.451	0.460	0.468
		NSEEN	**0.505**	**0.537**	**0.554**	**0.574**
	BioC2	DeepMatcher	0.373	0.463	0.491	0.517
		BioBERT	0.422	0.558	0.577	0.596
		USC–ISI	0.444	0.472	0.480	0.491
		NSEEN	**0.578**	**0.608**	**0.624**	**0.641**

We used the implementation published by the authors to perform our experiments. We used DeepMatcher with tuples containing only one field; the entity mention. We train DeepMatcher with the same initial pairs we use to train our model, and follow a common-word-based blocking technique recommended in their implementation to pre-process our data. DeepMatcher does not perform hard negative mining during its training, and the blocking is performed prior to the matching process in contrast to our framework.

4.4 Results

Table 3 shows the comparative results of our method (i.e., NSEEN) with other methods. We submit every query name in the BioCreative datasets to all systems, and retrieve the top k most probable IDs from each of them. We then find out if the correct ID (provided in the BioCreative dataset as labels) is present in the top k retrieved results (i.e., *Hits@k*) for several values of k. Our method outperforms the baselines in almost all settings. Chemical names are generally harder to normalize due to more sensitivity to parenthesis, commas, and dashes, but our method produces significantly better results.

Furthermore, Table 4 and the corresponding Fig. 4 show example protein name queries mapped to the UniProt reference set and the retrieved canonical names. Note that *none of the query names exist in the UniProt* reference set in the form provided as the query. Table 4 shows not only the syntactic variations

Table 4. UniProt sample queries and top-10 responses. The correct entities are indicated with a bold font and an asterisk. None of the queries have an exact string match in UniProt, and the lists include syntactically far correct responses.

S6K	PLCγ2	IKKϵ	H3
- **p70-S6K 1***	- **PLC-gamma-2***	- **IKK-epsilon***	- **Histone H3/a***
- p90-RSK 6	- PLC-gamma-1	- **IKKE***	- **Histone H3/o***
- **S6K1***	- **PLCG2***	- **I-kappa-B kinase epsilon***	- **Histone H3/m***
- **p70 S6KA***	- **Phospholipase C-gamma-2***	- **IkBKE***	- **Histone H3/b***
- S6K-beta	- Phospholipase C-gamma-1	- **IKBKE***	- **Histone H3/f***
- p70 S6KB	- PLC	- IKBE	- **HIST1H3C***
- 90 kDa ribosomal protein S6 kinase 6	- PLCG1	- IK1	- **Histone H3/k***
- 90 kDa ribosomal protein S6 kinase 5	- **Phosphoinositide phospholipase C-gamma-2***	- IK1	- **Histone H3/i***
- **52 kDa ribosomal protein S6 kinase***	- **PLC-IV***	- IKKG	- **HIST1H3G***
- RPS6KA6	- PLCB	- INKA1	- **Histone H3/d***

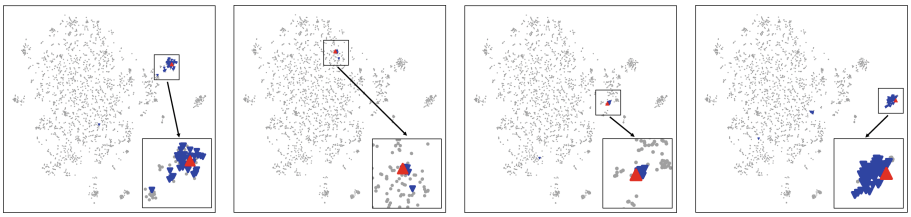
being captured by our method in the Top 10 responses, but the semantically equivalent names are included as well. These responses can have a significantly large string distance with the query name. e.g., (*S6K* $\longrightarrow$ *52 kDa ribosomal protein S6 kinase*), (*PLC*γ*2* $\longrightarrow$ *Phospholipase C-gamma-2*), (*IKK*ϵ $\longrightarrow$ *I-kappa-B kinase epsilon*), and (*H3* $\longrightarrow$ *Histone H3/a*).

Figure 4 sheds more light to the embedding space and highlights the same four query names and the names corresponding to the correct entities in the UniProt reference set. As shown in this figure most of the correct responses (in blue) are clustered around the query name (in red).

The retrieval time of the baseline methods are in the order of a few minutes. NSEEN relies on the approximate nearest neighbors architecture and provides highly competitive retrieval performance in the order of seconds. The study reported on [2] for approximate nearest neighbors architectures applies to our method as well.

5 Discussion

In this paper, we proposed a general deep neural network based framework for entity normalization. We showed how to encode semantic information hidden in a reference set, and how to incorporate potential syntactic variations in the numeric embedding space via training-pair generation. In this process we showed how contrastive loss can be used with non-binary labels to capture uncertainty.



(a) S6K (b) PLCγ2 (c) IKKϵ (d) H3

Fig. 4. tSNE representation of the example UniPort query entities shown in Table 4. Queries are red triangle and correct responses are blue. A sample of a thousand names from the reference set is shown with light grey dots to represent the embedding space. The bottom right insets show a zoomed version of the correct names clustered around the query name. (Best viewed in color) (Color figure online)

We further introduced a dynamic hard negative sampling method to refine the embeddings. Finally, by transforming the traditional task of entity normalization to a standard k-nearest neighbors problem in a numerical space, we showed how to employ a scalable representation for fast retrievals that is applicable in real-world scenarios without the need of traditional entity blocking methods. By eliminating the need for blocking as a pre-processing step, we can consider matches that are syntactically different but semantically relevant, which is not easily achievable via traditional entity normalization methods.

In our preliminary analysis, we experimented with different selection methods in the k-nearest neighbors retrieval process such as a *top-k majority vote* schema, but did not find them significantly effective in our setting. We also experimented with different soft labeling methods to dynamically re-rank the results such as *soft re-labeling the k-nearest neighbors*, but did not see much improvements to the overall performance. While currently highly effective, our method could benefit from improving some of its components in future research. We are also considering combining our approach with other embedding and collective reasoning methods to gain further potential performance improvements.

Acknowledgments. This work was supported in part by DARPA Big Mechanism program under contract number W911NF-14-1-0364.

References

1. University of Southern California - Information Science Institute Entity Grounding System (2018). http://dna.isi.edu:7100/
2. Annoy (approximate nearest neighbors oh yeah) (2019). https://github.com/spotify/annoy
3. Apweiler, R., et al.: UniProt: the universal protein knowledgebase. Nucleic Acids Res. **32**, D115–D119 (2004)
4. Arighi, C., et al.: Bio-ID track overview. In: Proceedings of the BioCreative VI Workshop (2017)

5. Bachrach, Y., et al.: Speeding up the Xbox recommender system using a euclidean transformation for inner-product spaces. In: Proceedings of the 8th ACM Conference on Recommender systems (2014)
6. Białecki, A., Muir, R., Ingersoll, G.: Apache Lucene 4. In: SIGIR 2012 Workshop on Open Source Information Retrieval (2012)
7. Cheatham, M., Hitzler, P.: String similarity metrics for ontology alignment. In: Alani, H., et al. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 294–309. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41338-4_19
8. Chen, H., Perozzi, B., Hu, Y., Skiena, S.: HARP: hierarchical representation learning for networks (2018)
9. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31164-2
10. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE TKDE **24**(9), 1537–1555 (2012)
11. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string metrics for matching names and records. In: KDD Workshop on Data Cleaning and Object Consolidation (2003)
12. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
13. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: a survey. IEEE TKDE **19**(1), 1–16 (2007)
14. Getoor, L., Machanavajjhala, A.: Entity resolution: theory, practice & open challenges. Proc. VLDB Endow. **5**(12), 2018–2019 (2012)
15. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2006)
16. Hastings, J., et al.: ChEBI in 2016: improved services and an expanding collection of metabolites. Nucleic Acids Res. **44**, D1214–D1219 (2015)
17. Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N.: Distributed representations of tuples for entity resolution. Proc. VLDB Endow. **11**(11), 1454–1467 (2018)
18. Jurczyk, P., Lu, J.J., Xiong, L., Cragan, J.D., Correa, A.: FRIL: a tool for comparative record linkage. In: American Medical Informatics Association (AMIA) Annual Symposium Proceedings (2008)
19. Kang, N., Singh, B., Afzal, Z., van Mulligen, E.M., Kors, J.A.: Using rule-based natural language processing to improve disease normalization in biomedical text. JAMIA **20**(5), 876–881 (2012)
20. Kotnis, B., Nastase, V.: Analysis of the impact of negative sampling on link prediction in knowledge graphs. In: WSDM 1st Workshop on Knowledge Base Construction, Reasoning and Mining (KBCOM) (2017)
21. Koudas, N., Sarawagi, S., Srivastava, D.: Record linkage: similarity measures and algorithms. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (2006)
22. Leaman, R., Islamaj Doğan, R., Lu, Z.: DNorm: disease name normalization with pairwise learning to rank. Bioinformatics **29**(22), 2909–2917 (2013)
23. Leaman, R., Lu, Z.: TaggerOne: joint named entity recognition and normalization with semi-Markov models. Bioinformatics **32**(18), 2839–2846 (2016)
24. Lee, J., et al.: BioBERT: pre-trained biomedical language representation model for biomedical text mining. arXiv preprint arXiv:1901.08746 (2019)

25. Mathew, J., Fakhraei, S., Ambite, J.L.: Biomedical named entity recognition via reference-set augmented bootstrapping. In: ICML Workshop on Computational Biology (2019)
26. Michelson, M., Knoblock, C.A.: Learning blocking schemes for record linkage. In: AAAI (2006)
27. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (2013)
28. Mudgal, S., et al.: Deep learning for entity matching: a design space exploration. In: Proceedings of the 2018 International Conference on Management of Data (2018)
29. Naidan, B., Boytsov, L.: Non-metric space library manual. arXiv preprint arXiv:1508.05470 (2015)
30. Neculoiu, P., Versteegh, M., Rotaru, M.: Learning text similarity with siamese recurrent networks. In: Proceedings the 1st Workshop on Representation Learning for NLP (2016)
31. Papadakis, G., Svirsky, J., Gal, A., Palpanas, T.: Comparative analysis of approximate blocking techniques for entity resolution. Proc. VLDB Endow. **9**(9), 684–695 (2016)
32. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
33. Peters, M.E., et al.: Deep contextualized word representations. In: Proceedings of NAACL (2018)
34. Ponomarenko, A., Avrelin, N., Naidan, B., Boytsov, L.: Comparative analysis of data structures for approximate nearest neighbor search. In: Data Analytics (2014)
35. Rastegari, M., Choi, J., Fakhraei, S., Hal, D., Davis, L.: Predictable dual-view hashing. In: International Conference on Machine Learning (ICML) (2013)
36. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
37. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: closing the gap to human-level performance in face verification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2014)
38. Yadav, V., Bethard, S.: A survey on recent advances in named entity recognition from deep learning models. In: Proceedings of the 27th International Conference on Computational Linguistics (2018)