Neural Model for Contextual Named Entity Retrieval

Sheikh Muhammad Sarwar
University of Massachusetts, Amherst
smsarwar@cs.umass.edu

Anna Fariha
University of Massachusetts, Amherst
afariha@cs.umass.edu

Abstract

In this project we propose CNER — a neural network model that exploits context of the entity for the named entity retrieval problem. The proposed model is designed based on multi-task learning paradigm where the main task is to discover entity similarity and the auxiliary task is to discover context similarity. We used weak supervision for context similarity with Siamese recurrent neural architecture, and showed that it is effective. Out hypothesis — "similar entities appear in similar contexts", was proved by the experimentation. We have also shown that the method supersedes a baseline in terms of recall@k.

1. Introduction

Consider a user reading news, who comes across the following text excerpt from a news article about the 2016 US presidential election:

> The last presidential candidate who lost an election despite winning the popular vote was Al Gore, and the similarities between them as candidates were always clear: intellectual introverts unable to connect emotionally with voters.

The user is interested in a list of candidates who lost an election despite winning the popular vote. To create a query, the user marks "Al Gore" as an exemplar from the list and provides the sentence as context. If we use a traditional list finding approach such as set expansion [5, 15, 9] we might find "Joe Biden" and "Dick Cheney" who were vice-presidents of United States. This approach would give high recall as it brings the names of all of the presidential candidates, but precision would be very low as the result set is not filtered according to context. Moreover, approach like set expansion only retrieves a list of entities without any explanation associated with them. There are cases where a user might have encountered an entity for the first time, and it is hard for her to assess the relevance of that entity without any evidence. We believe that if the system shows the user a sentence about the retrieved entity, it would be easier for her to determine relevance. Overall, considering context at the time of retrieval and displaying context at the time of presenting search results are both important and crucial for entity retrieval.

We refer to our task as *Contextual Named Entity Retrieval* (CNER) because we want to retrieve named entities (and their contexts) using only a single example entity and its context, where a context is the source sentence. We do not explore CNER approaches that make use of external resources like knowledge bases. We retrieve a set of potential sentences from a text collection and rank them by their scores toward finding the targets of interest. Our contribution to the project is as follows:

- We propose Contextual Named Entity Retrieval — a variant of classical entity retrieval.
- We construct a new data set for this task, and different neural architectures can be build on top of this data set.
- We frame the task as a *Multi-task Learning* problem, and propose a fully shared multi-task neural network for solving this problem.
- We used weak supervision for sentence similarity with recently proposed *Siamese Recurrent Neural Architecture* [17], and show that it is effective. Please note that in this paper we use the terms "context" and "sentence" interchangeably.
- We present the result of extensive experimentation to illustrate the effectiveness of the proposed model.

2. Related Work

As suggested above, work related to this study draws from many entity retrieval tasks, as well as general ranking tasks. We try to discuss related works from many domains and tasks and explain how our task contrasts.

2.1. List Completion and QA

List or set completion, list question answering (List QA), and entity ranking are similar to CNER in that their goal

is to generate a list of entities based on user need. CNER is different in that it supports an arbitrary entity type of ephemeral or limited interest and thus does not admit the use of prior knowledge or any other data sources.

2.1.1 List Question Answering (QA) Systems

List questions are a special class of questions, wherein a system has to come up with a set of entities in response to a question [21, 6]. Systems built for solving these tasks often leverage some form of well-trained entity recognition to collect the most promising answer passages [16, 13]. When the target type of a question is known in advance (e.g., *who?* suggests *person*), identifying passages with that type of entity helps narrow down the set of candidate results. In contrast, although CNER can leverage existing entity annotations as features, it assumes that the target type has no training data other than the starting query sentence.

2.1.2 Set Completion

In the list completion task, a small set of seed entities is given, with the aim of retrieving a complete set of target entities. Similar to our approach, Dalvi et al. [5] used a set expansion approach in their first stage to retrieve a set of entities given the seed entities, and then filtered those entities using an external source. However, the set completion task is also different from ours in a sense that it does not consider the context in which the entities appear and the disambiguation information it provides.

Recent works have introduced different kinds of contexts to set expansion. Zang et al. [23] proposed an extension of the existing row population functionality of spreadsheet applications, where rows were populated with entities given a set of seeds. Even though this task can be framed as set expansion, the authors had significantly different challenges as they considered column heading and table captions provided by users and made use of that information to find similar entities.

Hanafi et al. [12] proposed a framework that suggests extraction rules to a user given positive and negative examples of entities. Users can select and refine those rules for extracting entities more accurately. Oiwa et al. [18] also proposed user refinement of an entity list obtained against example entities that helped to find a target entity list with more interactions. Even though all of these tasks considered expansion from a set of seed entities, they included different information sources. In contrast, we propose CNER which does not require such additional information.

2.1.3 Entity Ranking Systems

Our proposed task is also closely related to the entity ranking task in INEX 2009 [8] and TREC 2010 [1]. Approaches

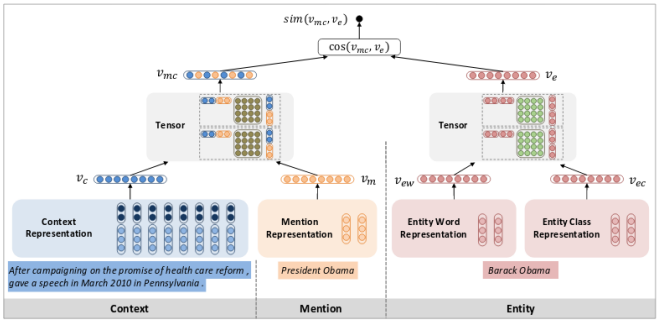

Figure 1: NN architecture for entity disambiguation (source: Sun et al. [19])

taken by participants in these tasks included the usage of co-occurrence models, NER based type filtering, and context modeling [9]. Co-occurrence modeling was further applied based on context i.e., relation between a pair of entities that was measured by their co-occurrence in documents (context-independent) and by computing similarity of the term vectors extracted from those documents (context-dependent) [4]. Some solutions augmented entity representation using entity synonyms and used search engines with that representation. Most of the participants (39 out of 48) of this task adopted external resources, while most of them were heavily dependent on Wikipedia for performing entity type filtering [9]. In this work, we model entity contextual information using vector representation of a sentence, but prohibit ourselves to make use of any external resources.

2.2. Entity Linking with Neural Networks

Our work is also closely related to named entity linking, that aims to link an entity in a document to the same entity in a knowledge base. There can be several variations of an entity mention and in order to link any two of those variations from document to knowledge base it is required to consider contextual information from document to build a precision oriented system. Recently proposed convolutional neural network architecture by Sun et al. [19] is a good example of the usage of a neural network for comparing two entities by computing similarities of their latent space representation computed by two different neural networks. The architecture is shown in Figure 1. However, this architecture does not allow any information sharing between these two models and allow to learn the representations jointly. Our proposed approach is inspired from this model, but it considers layer sharing that can improve over such an architecture.

3. Problem Statement and Data Set

We are given a collection of sentences, $S = \{S_1, \ldots, S_n\}$, and a query sentence S_q with an annotated entity set E_q in it. In our experiments, $S_q \in S$ but that is

not required. E_q will normally have one entity in it, but we allow the possibility that a sentence lists more than one name. Broadly speaking, our task is to define a function $F(S_i, S_q)$ that can score any sentence $S_i \in S$ based on the likelihood it contains a previously unseen entity of the same type as those in E_q. Here, the type of E_q is implicitly indicated by the contextual information present in the sentence.

To form our evaluation set, we begin with a set E of entities of the same type. Starting with any one entity in E, the task is to find all of the others. We form a query by selecting $E_q \in E$ and finding some sentence $S_q \in S$ such that E_q is mentioned in S_q, yielding a sentence that provides context for the entity. We additionally require S_q to be from a set of sentences that mention E_q that is consistent with the type of E rather than an unrelated mention of E_q.

The algorithm uses $F(\cdot)$ to rank $S_i \in S$. Let E_k be the union of E_q and the subset of E that has been seen in sentences at ranks 1 to k, with $E_0 = \{E_q\}$. We count a sentence at rank k as relevant only if $E_k \neq E_{1\ldots k-1}$ – that is, if a new target entity occurs. Relevance thus incorporates novelty in this evaluation.

Note that although we retrieve named entities in this task, target entities are of more finely-grained types compared to commonly explored named entities such as person, location, and organization. For example, if a query sentence contains information about the presidents of the United States and mentions the name of one or two of the presidents, our target is to find the rest of the sentences in the corpus that mention *other* presidents, not just to find other people. Moreover, our goal is to see the name of as many distinct presidents as possible in the top-k retrieved sentences. Even though we are retrieving person names in this case, *any* "person" is not sufficient. Any approach to this problem must infer the user's target entity need from the context of the sentence and find sentences that have a higher chance of having the correct context as well as unique and novel entities.

We adopt a curated version of the TREC 2005 and 2006 List Question Answering (QA) data set (see Table 1 for details), where a question is matched with a list of entities appearing in a set of documents. We break down those documents into sentences and disregard any document information associated with a sentence. Scoring a sentence independent of document information is a harder task, but it would help us to understand how informative a sentence is for this task.

For each list question we randomly pick a sentence that contains at least one entity for that question and find rest of the sentences that contains other similar types of entities. To illustrate the approach, consider this query sentence: "In 1986, **Susan Butcher**, the second woman to triumph in the Iditarod and a mushing superstar with three consecutive victories, arrived in Nome in 11 days 15 hours and six minutes for her first victory".

The query entity, E_q is the highlighted "Susan Butcher" mention and the task is to identify other Iditarod winners. (As with many such tasks there is an ambiguity inherent in the task: the goal could be female winners or mushing superstars.) Suppose that the following were the start of the ranked list of sentences:

1. in 1991, the first musher was **Rick Swenson**, who had already won the race four times but had been matched by *Susan Butcher*, who chalked up four relatively quick win as the second woman ever to win the race. (relevant)

2. the 1999 field included defending champ **Jeff King**, who have won the race three times; three-time winner **Martin Buser**; *Rick Swenson*, a five-time champion; and perennial front-runner DeeDee Jonrowe. (relevant)

3. three-time Iditarod winner *Martin Buser* of Big Lake, who placed second last year, said he is hopeful to avoid the injury that forced him to drop six dog early in the 1999 race. (non-relevant)

In each sentence, correct entities are marked in bold the first time they appear and italics in subsequent occurrences. The first two sentences contain target entities that had not been seen before. The third sentence only contains *Martin Buser*, already seen in the second sentence, so it is non-relevant.

Note that the second result of the ranked list contains a non-relevant entity, "DeeDee Jonrowe." Even though this entity is a person and is clearly of approximately the same type as the query entity, it was not explicitly annotated as relevant, so is treated as non-relevant text for the evaluation.

4. Technical Approach

In this section, we discuss the multi-tasking neural network model that we used to solve CNER. Our hypothesis is: *similar context would contain similar entities, and similar entities would be represented using similar context*. That's why we feed the query context and entity combined with the candidate context and entity into the same network. Before we start discussing our proposed model, we briefly mention the pre-trained word embedding that we have used for this task as well as the Siamese neural network model that we used for weak supervision.

4.1. Word Embedding

We build word embedding using the lemmatized form of the AQUAINT[1] text corpus. As there is an extensive discussion on the data sets in the experimental results section, we do not provide details on the corpus here. We use a recently proposed word embedding technique [2] that considers the

[1] http://www-nlpir.nist.gov/projects/aquaint/

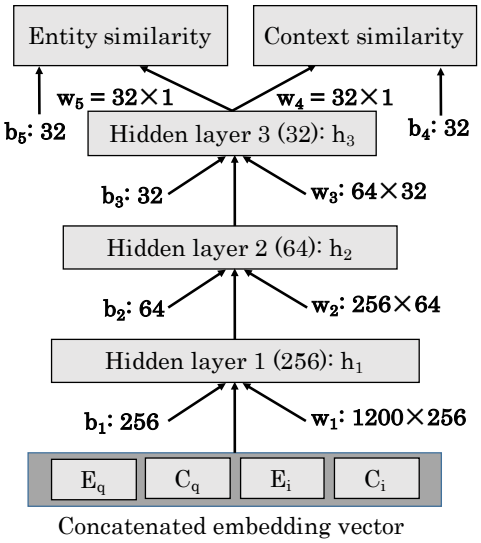

Figure 2: Architecture of fully connected multi-tasking neural model for CNER

morphology of words. This approach is based on the skip-gram model, where each word is represented as a bag of character n-grams [14].

We could have used pre-trained word embedding built from corpus like Wikipedia, but most of documents in AQUAINT are the news articles that were published by New York times before 2005. But, we also required the pre-trained word embedding computed from the Wikipedia corpus, as we used another data set to train our sentence similarity model. As a result, we decided to merge both word embedding files, but given two representations from two files for a single word, we kept the one computed from the AQUAINT corpus.

4.2. Network Architecture and Loss Function

The proposed network architecture is shown in Figure 2. The input to the network is a concatenation of the embedding of query entity E_q, query context $C_q = S_q - E_q$, candidate entity E_c, and candidate context $C_c = S_c - E_c$. These embedding vectors are computed using average of word embedding. For a single pair of query and candidate sentence we refer to this concatenated representation as x_n. As each of these representations is a 300-dimensional word embedding, we get a 1200-dimensional embedding vector as the input to the network. In the experimental results section, we describe how we create such a data set and obtain train/test/validation folds as required to carry out our tasks.

Our approach assumes full sharing i.e., hard parameter sharing among the tasks. To achieve this, we have concatenated the embedding and passed them through fully connected layers. Hidden layers' outputs are passed through ReLU non-linearities. The entity similarity and context similarity output layers use softmax non-linearity. The entity similarity and context similarity output of the network are $f^e(x_n, \theta)$ and $f^c(x_n, \theta)$, respectively. The ground truth judgment for i^{th} pair of entities is y_i^e, and the weak supervised similarity value for i^{th} pair of contexts are y_i^c. There is a following section which describes how we compute context/sentence similarity using another neural network (weak supervisor network), as we do not have context similarity values from our data set. Our multi-task network eventually tries to learn that sentence similarity value y_i^c, provided by the weak supervisor network.

One loss component, L_{ce} is measured using cross-entropy after the computation of softmax non-linearity from the entity similarity output. Another loss component is measured using Mean Squared Error (MSE) between the softmax output from the network and y_n^c. Our assumption is entity similarity and context similarity both would help to find relevant entities and we define our complete loss function using Equation 1 by combining both loss components. Here, α is a hyperparameter and it is optimized on the validation set. Value of α indicates how much importance is given to each component of the loss function. In ideal multi-tasking case equal importance should be given to both of the tasks, but best validation accuracy might be obtained for any value of α. It is to be noted that here our goal is to obtain the best accuracy only for the entity similarity task. Sentence similarity has been used here as an auxiliary task to aid the entity similarity task.

$$\theta_* = \underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} \alpha L_{ce}(y_n^e, f^e(x_n, \theta)) + (1-\alpha) \mid\mid y_n^c - f^c(x_n, \theta) \mid\mid_2^2 \quad (1)$$

A recent architecture for entity disambiguation task computed separate representations of query and candidate entities and computed similarities using those representations [20]. Deviating slightly from that idea, we combine the query and candidate representations using a fully connected shared space.

4.3. Weak Supervision for Sentence Similarity

From our engineered data set we know which entities are relevant and similar given any random entity, but we do not know about context or sentence relevance. However, our goal is to learn entity and context (which is a sentence) similarity jointly. So, we use an external sentence similarity computation model and train our network to learn sentence similarity values produced by that model. This technique is referred to as weak supervision and recently this approach was shown to be effective for ranking documents [7]. Section 4.2 discusses on how we use context and contextual similarity as an auxiliary task to improve the entity similarity

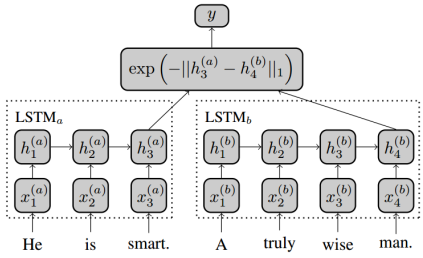

Figure 3: Siamese recurrent architecture for learning text similarity (source: Mueller et al. [17])

task. We use Siamese recurrent architecture which is also a neural model as a weak supervisor for learning context similarity.

Siamese recurrent architecture was introduced by Mueller et al. for computing similarity between short text like sentences [17]. The architecture is composed of two LSTM units, where each of them is used to represent one sentence. After obtaining two representations from two networks those are compared using Manhattan distance. The architecture is shown in Figure 3.

4.4. Evaluation Metrics

For a query sentence with an entity in it, we get a ranked list of candidate sentences according to the score computed by our network. A sentence at position k in the ranked list is considered relevant if it contains a relevant entity that has not been observed among the previous $k-1$ retrieved sentences. Thus, a relevant sentence must contain at least one unique relevant entity. Based on this notion of relevance we compute the standard information retrieval evaluation metrics — precision@k and recall@k, on the ranked list sorted by the scores produced by our proposed network.

Precision@k: The notation @k denotes the set of items in the top-k results. Precision@k is defined by Equation 2.

$$precision@k = \frac{\text{\# recommended items @k that are relevant}}{\text{\# recommended items @k}} \tag{2}$$

Recall@k: Recall@k is defined by Equation 3.

$$recall@k = \frac{\text{\# recommended items @k that are relevant}}{\text{total \# relevant items}} \tag{3}$$

5. Experimental Setup

In this section we provide a discussion on our experimental setup for computing word embedding, generating sentence similarity scores from the Siamese recurrent architecture, and finally computing the entity similarity score using our proposed multi-task learning neural model.

Table 1: Data set statistics

Number of Queries	120
Avg. Number of Entities Per Query	9.09
Avg. Number of Candidate Sentences Per Query	25,229
Avg. Number of Sentences with at least one ground truth entity	165
Avg. Number of Entity Tokens	409
Avg. Entity Density $= \frac{\text{Sentence with an entity}}{\text{Number of sentences}}$	0.8%
Avg. Number of TREC-judged Sentences	34
Avg. Number of Sentences Annotated with Entities Found from Judged sentences	165

5.1. Data set Curation

In this project, we contribute a new data set for continuing further experiment on this task. The data set is taken from TREC 2005 and TREC 2006 List Question Answering (QA) track (Data set description is given in Table 1). Even though this is a question answering track, we curated to suit our task. These two tracks had more than 200 questions altogether, and for each of these questions the answers were a list of entities. We filtered out questions for which answer list construction was easy (for example: list of countries). Finally, we ended up with 120 questions.

Against each question in list QA, along with entity list, the data set contains a set of documents, where the answers to that list question appears. Those documents were found by searching AQUAINT corpus with the list questions using a keyword based search engine. Against a list question, at first we tagged all documents with the answers to that list question. As a result, we obtained a set of documents where a group of similar entities appear. Then we broke those documents into sentences and removed all document related information from those sentences. After that our task is to pick a random sentence that contains an answer entity from the newly constructed collection of sentences (S), consider it as a query q, and then score all other sentences based on their likelihood of containing a similar entity.

For each of the 120 list questions, we pick a random sentence as query sentence, and consider other sentences as candidate sentences. We had around 29000 candidate sentences on an average against a single query, and very few of them contained entities similar to query entity. Learning with such a huge, imbalanced and sparse data set is difficult. Hence, we used BM25 search to create a pool of 2000 candidate sentences against each query sentence.

Each of our query sentence q, contains at least one entity E_q of the target type. From q we remove all the tokens that are part of target entity E_q, and obtain context C_q. We also use Stanford NER tagger [10] to determine the type of query entity (Person, Location, Organization, Time, Money etc.). Then, we also use the same tagger to find entities of similar type from candidate sentence. For each similar type of entity, we consider it as the entity to measure similarity with E_i, and consider the rest part of the sentence as context C_i. As a candidate sentence may contain more than one similar entities, we can form several training data cases from a single pair of sentences.

As for an example, let's consider that our query sentence is "**UMass CICS** is a great place for learning computer science", and the candidate sentence is "**UMass Boston** computer science department is funded by State Office of Massachusetts". In the query, entity of interest is UMass CICS, and in the candidate, the entity of interest is UMass Boston. The query entity indicates that entity of interest is organization. Considering this we have two candidate entities in candidate sentence: **UMass Boston** and **State Office of Massachusetts**. Then we create the following two training data cases:

• ('UMass CICS',"is a great place for learning computer science", "UMass Boston", "computer science department is funded by State Office of Massachusetts") $\rightarrow$ 1.

• ('UMass CICS', "is a great place for learning computer science", "State Office of Massachusetts", "UMass Boston computer science department is funded by") $\rightarrow$ 0.

We can see that the first training data case is relevant, where the second training data case is irrelevant, because State Office of Massachusetts is not our target entity, considering that the user is looking for universities in Massachusetts.

By applying the above technique, we can create a large number of training data cases, that are suitable for neural models. At the time of evaluation, we consider each entity in a test sentence, create (entity, context) pairs from that sentence, score them and select the score of the pair that achieved maximum score. Then we set that score as the score of the candidate test sentence.

5.2. Word Embedding Computation

We build word embedding using the lemmatized form of the AQUAINT text corpus [11]. Using the `Fast Text` tool by Bojanowski et al. [2], we derived 300 dimensional skipgram embedding with a context window of 5, no negative samples, and the recommended sampling threshold of 10^{-5}. The raw text of the AQUAINT corpus is 1.6 GB and it contains 538M words.

It is possible to set the `minCount` parameter, before running the fast text script. It indicates the minimum number of times a word must occur to be a candidate for which embedding will be computed. We explored with `minCount=3` and `minCount=5`, and found that the final vocabulary size was 798567 and 507865. Finally, we selected the second setting, because it was producing a reasonable number of words which, along with their embedding, could be loaded into memory.

5.3. Context Similarity Computation Using Siamese Neural Network

In Section 4.3, we discussed the architecture of Siamese recurrent neural network (SRNN). We used this network as we did not have ground truth judgments in our data set for sentence level similarity. We trained SRNN on Stanford SNLI data set [3] that comes up with binary judgments from human for a pair of sentences. This data set contains 367372 pairs of sentences and it is a decent number of training examples for training a sentence similarity model. The test data contains 36738 pairs of sentences. To get the embedding for the words in the SNLI corpus we used the pre-trained word embedding file build on Wikipedia using Fast Text [2]. We used the same technique to compute embedding for words in AQUAINT corpus. We could have used Wikipedia embedding, but by doing that we might have missed embedding for a lot of words in the AQUAINT corpus. The parameter configuration used to train SRNN are is in Table 3. We obtained 79.1% accuracy on the test data with this configuration. We were not particularly interested to increase its accuracy, as the labels generated by them were used for weak supervision, and we did not want our multi-task network to over-fit on these labels.

5.4. Multi-task Network Configuration for Sentence Scoring

The Multi-task network proposed in this project is shown in Figure 2. It receives a 1200 dimensional vector constructed using the concatenation of the embedding vectors from query entity, query context, candidate entity and candidate context. We trained this network for 40 different hyper parameter configurations. For each configuration, it took around 3 hours for the network to run, and we ran them in parallel using gypsum cluster. Data sizes for training, validation, and test fold was 268200, 16890, and 78920, respectively.

6. Experimental Results

In this section, we discuss the performance of multi-task learning and compare it with a very standard baseline sentence embedding with average of word embedding. In the previous section we mentioned that we have tried to improve the system with a range of parameters. Table 4 shows the performance of our proposed approach for the top-10 parameter setting. The table shows the performance of proposed

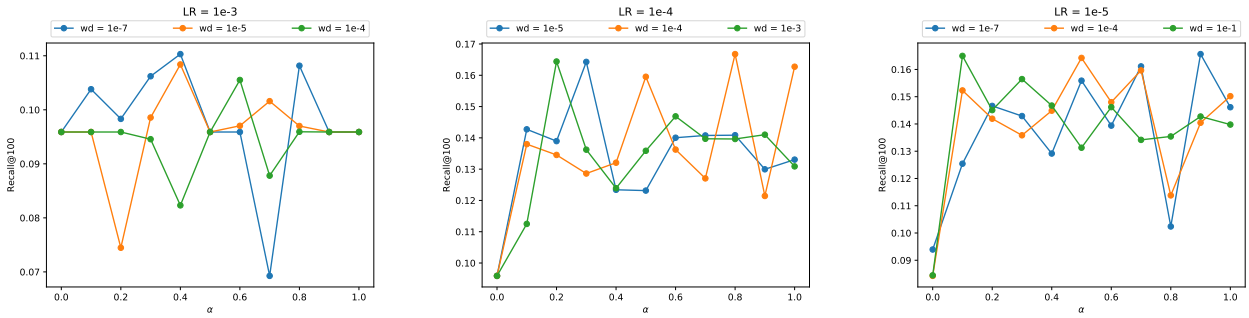

Table 2: Recall@10 for different values of hyper parameters

Table 3: Hyperparameter configuration for training SRNN

Learning Rate	1e-3
Dropout Keep Probability	0.9
L2 Regularization Strength	1e-4
Hidden Units	50
Batch Size	128
Number of Epochs	100
Optimizer	Adam

approach using recall@100 evaluation metric. Even though the recall value is not high compared to some other tasks, we attribute this to repeated entities. There are many entities in the corpus that appear multiple times, and increases the number of relevant sentences. If the model misses one of these entities, it has a higher chance to miss the rest. Even though our problem is focused towards entity uniqueness, we could not find a better way to incorporate into the loss function, and hence we do not show results for unique entities.

Table 4 also shows that multi-task learning is somewhat effective. The best recall can be achieved by the setting the value of α to 0.8. Please refer to our loss function in Section

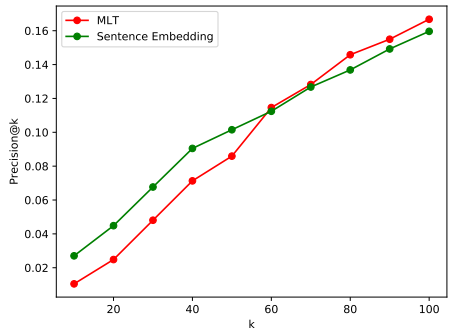

Figure 4: Recall@k for different values of k for both MLT and sentence embedding approach

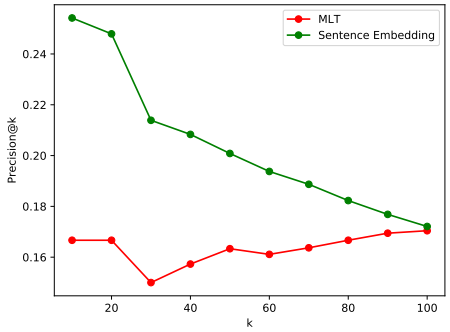

Figure 5: Precision@k for different values of k for both MLT and sentence embedding approach

Table 4: Top 10 recall@100 for different hyper parameter settings

Learning Rate	Weight decay	α	Recall@100
0.0001	0.0001	0.8	0.168
1e-05	1e-07	0.9	0.166
1e-05	0.1	0.1	0.164
0.0001	0.001	0.2	0.164
0.0001	1e-05	0.3	0.164
1e-05	0.0001	0.5	0.164
0.0001	0.0001	1.0	0.162
0.0001	1e-07	0.2	0.161
1e-05	1e-07	0.7	0.161
1e-05	1e-08	0.7	0.160

4.2 for a detailed discussion on the loss function. However, the results indicate that it is still very important to solve the entity similarity task, and the auxiliary sentence similarity task is not contributing heavily towards solving it. Figures in Table 2 is also telling the almost the same thing, but with a different representation. It is important to find a suitable value of α, and it should be treated as a hyperparameter that requires optimization.

Figure 4 and 5 shows the comparison of our proposed approach with sentence embedding that is computed using

average of word embedding. Using sentence embedding we compute the representation of the query as well as the candidate sentence and compare them using cosine similarity. After that we rank sentences using the similarity value. The figures show that increasing recall has less effect on the precision of our system. Sentence embedding faces drastic decrease in precision at higher ranks. However, overall we can only achieve gain in recall at higher ranks. This is desirable considering the task, as users would be interested to find as many relevant entities as possible, even if they have to go through a large list of sentences. Nonetheless, we believe that replacing the fully shared architecture with a more optimized one will bring improvement in the earlier ranks of retrieval. Even though sentence embedding seems like a very simple method, in some tasks it has shown improvement over the state-of-the-art LSTM models [22].

7. Conclusions

In this project, we propose Contextual Named Entity Retrieval (CNER), and propose a multi-task learning solution for it. Our hypothesis - "similar entities appear in similar context", was proved by the experimentation that we carried out. We compared our results by computing query and candidate representation using average of word embedding and showed that our method is competitive with the baseline. Average of word embedding was successful in different NLP task compared to the LSTM, and it is hard to improve over this baseline. In this project, we used a fully shared multi-task network, which often hurts performance as it shared layers might end up solving only one task. In future, we would like to examine different sharing schemes for Multi-task learning and try to obtain better performance.

References

[1] K. Balog, P. Serdyukov, and A. P. d. Vries. Overview of the TREC 2010 entity track. Technical report, DTIC Document, 2010.

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[3] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *EMNLP*. Association for Computational Linguistics, 2015.

[4] M. Bron, K. Balog, and M. d. Rijke. Related entity finding based on co-occurrence. Technical report, AMSTERDAM UNIV (NETHERLANDS), 2009.

[5] B. B. Dalvi, J. Callan, and W. W. Cohen. Entity list completion using set expansion techniques. In *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010*, 2010.

[6] H. T. Dang, J. Lin, and D. Kelly. Overview of the TREC 2005 Question Answering Track. In *TREC 2006*, 2000.

[7] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. Neural ranking models with weak supervision. SIGIR '17, New York, NY, USA.

[8] G. Demartini, T. Iofciu, and A. P. De Vries. Overview of the inex 2009 entity ranking track. In *Proceedings of the Focused Retrieval and Evaluation, and 8th International Conference on Initiative for the Evaluation of XML Retrieval*, INEX'09, pages 254–264, 2010.

[9] Y. Fang and L. Si. Related entity finding by unified probabilistic models. *World Wide Web*, 18(3):521–543, May 2015.

[10] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370. Association for Computational Linguistics, 2005.

[11] D. Graff. The AQUAINT Corpus of English News Text. CDROM, 2002.

[12] M. F. Hanafi, A. Abouzied, L. Chiticariu, and Y. Li. Seer: Auto-generating information extraction rules from user-specified examples. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6672–6682, New York, NY, USA.

[13] C. Lee, Y.-G. Hwang, H.-J. Oh, S. Lim, J. Heo, C.-H. Lee, H.-J. Kim, J.-H. Wang, and M.-G. Jang. Fine-grained named entity recognition using conditional random fields for question answering. In *Asia Information Retrieval Symposium*, pages 581–587. Springer, 2006.

[14] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

[15] D. Moldovan, C. Clark, and M. Bowden. Lymbas poweranswer 4 in trec 2007. 2007.

[16] D. Mollá, M. Van Zaanen, D. Smith, et al. Named entity recognition for question answering. 2006.

[17] J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792, 2016.

[18] H. Oiwa, Y. Suhara, J. Komiya, and A. Lopatenko. A lightweight front-end tool for interactive entity population. *CoRR*, abs/1708.00481, 2017.

[19] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1333–1339. AAAI Press, 2015.

[20] Y. Sun, L. Lin, D. Tang, N. Yang, Z. Ji, and X. Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*, pages 1333–1339, 2015.

[21] E. M. Voorhees and H. T. Dang. Overview of the TREC 2005 Question Answering Track. In *TREC 2005*, 1999.

[22] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198, 2015.

[23] S. Zhang and K. Balog. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 255–264, New York, NY, USA. ACM.