TREMA-UNH at CAR 2019

Jordan Ramsdell, Sumanta Kashyapi, Shubham Chatterjee, Pooja Oza, Laura Dietz

{jsc57, sk1105, sc1242, pho1003}@wildcats.unh.edu, dietz@cs.unh.edu

TREMA lab, University of New Hampshire, U.S.A

Abstract

This notebook describes the submissions of team TREMA-UNH to the TREC Complex Answer Retrieval, TREC News, TREC Conversational Assistance, and TREC Deep Learning tracks in 2019. We explore passage retrieval systems, passage similarity metrics, and neural network methods that address the task statements of these tracks.

1 Introduction

This year, team TREMA-UNH from the University of New Hampshire, USA, participated in the following TREC tracks described as follows:

- **Complex Answer Retrieval**: Retrieval of passages that are relevant to a given topic, and then ordering these passages into a comprehensive article.
- **Conversational Assistance**: Using dialogue and its context, retrieve and rank passages from a large corpus that satisfy the informational needs of the dialogue.
- **News**: Given a news article and a corpus of entities, retrieve and rank entities that are relevant to the content of the news article.
- **Deep Learning**: Information retrieval with a large training data set. Primarily focused on neural network methods.

2 Complex Answer Retrieval

Answers to difficult questions can be complex, requiring that the user understands multiple topics addressed by the answer. The information needs of a user may not be by traditional retrieval methods: giving a user a ranked list of documents relevant to their question does not mean that the user will understand the answer. For example, some documents may logically precede others, such as those that introduce a topic that is expanded on by later documents. If a user reads these documents out of order, then the answer can be difficult to comprehend. There is a need, then, for answers structured in a way that is easy for a user to understand.

Wikipedia pages provide a good example of structured information: the page is divided into multiple sections that address topics relevant to understanding the subject of the page. Passages are logically ordered in each section such that they intro and then expand on the topic of the section. Using structured information can make the task of finding relevant information easier for the user to determine which topics are relevant to their question, and only read documents pertaining to these topics.

Our work in the Complex Answer Retrieval track uses structured information to address the following task.

Task. Given a topic and an outline consisting of section headings, retrieve up to 20 passages and organize them in a topically coherent way.

2.1 UNH-bm25-ecmpsg

For each topic, T, we retrieve the top 1000 passages, P_T, that are relevant to the topic with respect to the Lucene's[1] implementation of the BM25 metric. These passages contain entities (links to Wikipedia), and we denote E_{PT} as the set of all entities contained in passages retrieved for the given topic. We measure the relevance of an entity $e_i \in E_{PT}$ to the topic T using the set of passages, P_{e_i} in P_T that link to entity e_i, and the following formula:

$$\text{Relevance}_T(e_i) = \sum_{p_j \in P_{e_i}} \text{BM25}_T(p_j)$$

Where $\text{BM25}_T(p_j)$ is the relevance score of a passage with respect to a topic using BM25. Therefore, an entity's relevance score is equal to the sum of the BM25 scores of all the passages in the candidate set that link to the entity.

We take the top 100 entities with respect to their relevance to a topic and expand the initial BM25 passage query with the names of these entities. Using the expanded query and BM25, produce a ranking of passages given a topic. We convert this ranking to an ordering of 20 passages using the conversion script[2] provided by TREC CAR Y3.

2.2 Reordering Passage Rankings Based on Similarity

In this section, we describe methods of reordering a candidate set of passages retrieved from section path queries. We reorder candidate sets of passages based on a passage similarity metric, such that passages that are similar to each other are grouped together.

[1]Lucene can be found at https://lucene.apache.org/

[2]The script can be found at https://github.com/TREMA-UNH/car-convert-ranking-to-ordering

2.2.1 Generating Candidate Passage Sets

We form each section path query by concatenating the name of the topic to each section header present in the outlines. In TREC CAR Y3, we are limited to retrieving up to 20 passages per topic, and so our candidate sets of passages consist of the top n passages from each section query, where $n = \frac{20}{\text{\# of sections}}$. When this results in fewer than 20 passages, we randomly pick the remaining passages from the section path queries

2.2.2 Reordering Candidate Passages

For each candidate set of passages, we begin by randomly picking a passage from the candidate set, which we denote the seed passage. Using a passage similarity metric, we then find the passage in the remaining candidate set that is most similar to the seed passage and place it next in the ordering, where it becomes the new seed passage. We repeat this process, retrieving the next passage that is most similar to the current seed passage, until we have ordered all 20 passages from the candidate set.

2.2.3 Passage Similarity Metrics

We use the following passage similarity metrics to reorder the candidate passage sets. These metrics are named according to the names of the runs submitted to TREC CAR Y3 that utilize them.

2.2.4 UNH-tfidf-[stem/lem/ptsim]

The UNH-tfidf passage similarity metric represents passages as vectors using the bag-of-words model. Each component of the vector is a unique term contained in the passage, the coefficients of which are equal to the term-frequency-inverse document frequency of each unique term. We then calculate the similarity between a pair of passages using cosine similarity based on their vector representations:

$$\cos(p_i, p_j) = \frac{\vec{p_i} \cdot \vec{p_j}}{||\vec{p_i}||||\vec{p_j}||}$$

We evaluate the TFIDF cosine similarity metric with respect to three methods of pre-processing paragraph text:

- **stem:** Passages in the corpus are first stemmed using Lucene's English Analyzer.
- **lem:** Passages in the corpus are first lemmatized.
- **ptsim:** Passages in the corpus are not pre-processed: only the raw terms are used.

2.2.5 UNH-bm25-[stem/lem]

Our UNH-bm25 method uses Lucene's implementation of BM25 as a passage similarity metric. Let (p_i, p_j) be a pair of passages from the paragraph corpus. Then BM25(p_i, p_j) is the BM25 score of passage p_j with respect to treating the text contained in p_i as the query. Note however that BM25 is not a symmetric metric. We construct the BM25 passage similarity metric by making BM25 symmetrical: $\text{BM25}_{\text{sim}}(p_i, p_j) = \frac{\text{BM25}(p_i,p_j)+\text{BM25}(p_j,p_i)}{2}$

We evaluate the UNH-bm25 similarity metric with respect to two methods of pre-processing paragraph text:

- **stem:** Passages in the corpus are first stemmed using Lucene's English Analyzer.
- **lem:** Passages in the corpus are first lemmatized.

2.2.6 UNH-dl[layer size]

We use the similarity score obtained from Section 3.1.1 to reorder the passages. Based on the layer size of the dense layers used in the model we have two variants: dl100 and dl300.

2.3 UNH-ecn

We start with an entity and passage run. For every query-entity pair, we create an *Entity Context Document*(ECD). To construct this ECD, we filter all passages which mention the entity and "stitch" them together into one "document" about the entity. All entities which occur in this ECD co-occur with the target entity (the entity the ECD is about). We derive a distribution over these co-occurring entities by using the frequency of these entities, that is, the number of times the entity occurs in the ECD. For every passage in an ECD, its score is equal to the sum of the frequency scores of the entities in the passage. We rank the passages using this score. This gives us a passage ranking for every query-entity pair. We call this ranking as a support passage ranking. We obtain a passage ranking from this support passage ranking by marginalizing over the entities. We convert this ranking to an ordering of 20 passages using the conversion script provided by TREC CAR Y3.

2.4 UNH-qee

As in Section 2.3, we obtain a distribution over the co-occurring entities with a given entity using the frequency of occurrence of these entities. In this method, we rank the co-occurring entities using this score. We use the top 20 entities from this ranking to expand the query and retrieve passages with the expanded query using BM25. As in Section 2.3, this gives us a support passage ranking. We obtain a passage ranking from this support passage ranking by marginalizing over the entities. We convert this ranking to an ordering of 20 passage using the conversion script provided by TREC CAR Y3.

2.5 UNH-neural

In this method, we construct a neural network to score the relevance of passages with respect to a query. We construct an embedding of each

Table 1: Results of TREC CAR Y3 methods, where facet overlap and relevance are the evaluations metrics measured using for evaluating submissions. Bold values indicate the method with the best performance with respect to an evaluation metric. Asterisks indicate methods where there is no statistically significant difference to the best method with respect to standard deviation.

Method	**Facet Overlap**	**Relevance**
UNH-bm25-ecmpsg	0.0295 ± 0.0065*	0.0931 ± 0.0175
UNH-bm25-rm	0.0658 ± 0.0114*	**0.1297 ± 0.0170***
UNH-bm25-stem	0.0622 ± 0.0096*	0.1141 ± 0.0165
UNH-dl100	0.0403 ± 0.0072	0.1134 ± 0.0165*
UNH-dl300	0.0335 ± 0.0065	0.1093 ± 0.0159*
UNH-ecn	0.0016 ± 0.0010	0.0188 ± 0.0040
UNH-neural	0.0295 ± 0.0065	0.0931 ± 0.0139
UNH-qee	0.0427 ± 0.0079	0.1201 ± 0.0162*
UNH-tfidf-lem	0.0686 ± 0.0105*	0.1150 ± 0.0165*
UNH-tfidf-ptsim	**0.0756 ± 0.0115***	0.1230 ± 0.0174*
UNH-tfidf-stem	0.0674 ± 0.0105*	0.1168 ± 0.0165*

passage using ELMo [1]. ELMo produces an embedded word vector for each word in a sentence based on the word's context in the sentence. We construct a sentence embedding by taking the mean of the word vectors in a sentence, and a passage embedding by taking the mean of the sentence embeddings contained in the passage. We also embed each query via ELMo by treating it as a sentence. In addition to passage and query embeddings, we create a passage relevance vector for each passage that represents the relevance of the passage given the query. Each element of the passage relevance vector corresponds to the inverse rank score of the paragraph, with respect to the query, under a particular passage retrieval system (for example, BM25).

The input layer of our neural network is a fully connected linear layer, in which the passage embedding, passage relevance, and query embedding vectors are mapped onto three vectors of length 100. We use tanh as the activation function for this layer. This is used as the input of the second layer, which is a weighted trilinear product between the three vectors. We then apply the logistic function to the output of the trilinear function.

We train the neural network using logistic regression: passages are labeled with a 1 if they are relevant with respect to a query, or 0 otherwise, according to the TREC CAR Y1 train section-level passage qrels. Once trained, we use the neural network to rank a candidate set of passage for each query (retrieved using BM25). We convert these rankings to an ordering of 20 passages using the conversion script provided by TREC CAR Y3.

2.6 Results from the CAR Evaluation

We use the following two metrics to evaluate the performance of our TREC CAR Y3 submissions.

Facet Overlap. For an ordering of 20 passages retrieved for a topic and a section outline, we define facet overlap as the number of transitions between passages that are in the same section over the total number of transitions between passages. We obtain the final score by averaging over the facet overlap scores across all topics in Trec Car Y3 Test.

Relevance. We define relevance as the number of relevant passages retrieved over the total number of passages retrieved for a topic. We determine relevance of a passage to a topic by using the Trec Car Y3 Test qrels. We obtain the final score by averaging over the relevance scores across all topics in Trec Car Y3 Test.

Table 1 shows our results for TREC CAR Y3 with respect to the facet overlap and relevance evaluation metrics. We see that UNH-tfidf-ptsim performs best with respect to the facet overlap metric, and that UNH-bm25-rm is the best method with respect to the relevance metric. However, these methods are not significantly better than many of the other methods (marked with asterisks in Table 1) with respect to the standard deviation of the metrics. This discrepancy is most likely due to the small number of topics (55 in total from Y3 Test) used for evaluation.

We also see that UNH-tfidf-ptsim is significantly better than UNH-bm25-ecmpsg with respect to facet overlap. This is noteworthy because the UNH-tfidf-ptsim method uses UNH-bm25-ecmpsg to retrieve a candidate set of passage, and then reorders the passages based on the TFIDF cosine similarity metric (see sections 2.2.2 and 2.2.4).

2.7 Conclusion

Retrieving and rankings passages based on relevance to a topic does not necessarily produce an understandable summary. We assume that if a topic has multiple aspects (such those described by section headers in a page), that passages of the same aspect should be grouped together, making it easier for the user to identify what aspects are relevant to their question. Our passage similarity methods address this by reordering retrieved passages in such a way that similar passages are grouped together, under the assumption that similar passages belong to the same aspect. We demonstrate that this can directly improve an existing passage retrieval system (in particular, UNH-bm25-ecmpsg) with respect to our facet overlap and relevance evaluation metrics. Furthermore, these methods can work with any passage retrieval system, implying that they can improve other passage retrieval systems. In the future, we hope to find better passage similarity metrics for use in reordering passages: we expect that such metrics would drastically improve the results seen in our paper.

3 Deep Learning

This track studies how Information Retrieval can benefit from large training data and which methods in particular perform well in this setting. The track has two different tasks each with two different subtasks: Document ranking and Passage ranking and their corresponding full-ranking and re-ranking subtask. However, we only participate in passage re-ranking task.

Task Given an initial ranking of 1000 passages, we have to re-rank these passages based on their likelihood of containing an answer to the question.

3.1 Methods

3.1.1 Siamese neural model with ELMo embeddings

Siamese neural architecture is a family of neural architecture for which there exists at least one pair of layer which are identical or in other words share the same parameter values. Our intuition is if two passages are similar in the context of belonging to the same Wikipedia section, then there exists an embedding space where their corresponding representation will be closer than other passage pairs which are not similar. The job of any model that attempts to learn the similarity metric discussed here has to learn this embedding space. Siamese network allows us to look at both passage representation in a pair of input data sample at once. Hence for any kind of pairwise similarity modeling, it is natural for a siamese network to learn an embedding space which projects similar data points close by and dissimilar data points far apart. Also as both of the input layer in a siamese network share the learned parameters the resulting model will be symmetric which means it will produce the same output even if the order of two passages in a pair is switched.

Detailed architecture

Figure 1 depicts the detailed architecture which is used for our experiments. The model accepts a pair of passages from the training set in form of ELMo vectors. They are fed to two siamese dense layers (share learned parameters), $DL1_a, DL1_b$ and $DL2_a, DL2_b$. The output from these layers are concatenated and fed to another dense layer, $DL3$. Finally its output is fed to $DL4$ which yields the output for the model. Table **??** gives the details of each layers in terms of tunable parameters. We tried various combinations of layer numbers and sizes but found the setting described here to be most effective in terms of validation loss and convergence time.

Training For training we use passage representations of 80% samples of balanced y1 train parapair dataset and use rest of it as validation set. For passage representation we use concatenated ELMo vectors. For example let a sample in our training dataset is the passage pair (p_1, p_2). We obtain 3 layers of ELMo vectors for each passage $([E^1_{p_1}, E^2_{p_1}, E^3_{p_1}], [E^1_{p_2}, E^2_{p_2}, E^3_{p_2}])$ each of length 1024 and concatenate for each passage to obtain (E_{p_1}, E_{p_2}) where E is the concatenation of the three ELMo layer vectors E^1, E^2, E^3. Hence each passage representation is a vector of length 3072.

Reducing overfitting To reduce overfitting in training we use regularization, dropout and early stopping. The regularization for each layer in the model is set to 0.0001 and one dropout layer is introduced for each input layer with a dropout rate of 0.5 to prevent overfitting. We also employ early stopping to stop the training some iterations after the vali-

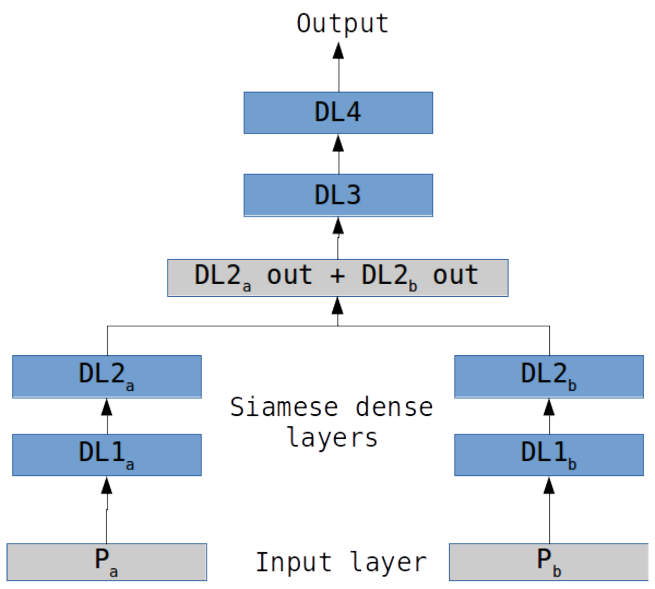

Figure 1: Siamese architecture

Table 2: Results of the methods submitted for DL track

Method	**AP**	**mean NDCG**	**mean P@10**
UNH-bm25	0.2565	0.5546	0.3465
UNH-exDL-bm25	0.0364	0.14	0.0605

dation f1 score stops increasing. The performance of the resulting model is measured in terms of AUC score and the resulting best model is used for the TRECCAR task.

3.1.2 Query expansion using Siamese DL model

As discussed in the previous section, we model a similarity metric using siamese neural model. We use this similarity metric to find the most similar passage from the TRECCAR benchmark Y1 dataset and use it to expand the query. Then we use the expanded query to retrieve passage ranking. We refer to this method as UNH-exDL-bm25.

3.2 Results

We submit runs obtained from two of our methods: UNH-bm25 (BM25 baseline method) and UNH-exDL-bm25. Results obtained for our methods are described in Table **??**. The average of the median of all the submissions are the following: AP 0.3864, NDCG 0.6457 and P@10 0.5651.

3.3 Conclusion

We observe that our query expansion model performs poorly on the re-ranking task. This suggests that our choice of knowledge base for the query expansion (TRECCAR benchmark Y1) is not suitable for the task.

4 Conversational Assistance

Task. The task of Conversational Assistance is to retrieve the passages using contextual information provided by series of queries on a given topic. Three automatic runs were submitted.

4.1 Methods

UNH-trema-rel For each query, we retrieve feedback passages P by using BM25. We then generate a candidate entity list E which consists of all the entity mentions present in the feedback passages. For every entity e_i of the candidate entity list E, we create an entity-pair (e_i, e_j) with every other entity of the candidate list. For every entity-pair (e_i, e_j) we check the presence of both entities e_i and e_j in passage. If the entity-pair is present in the passage, then the score of the entity-pair is:

$$f_{ecr}(e_i, e_j) = \sum_{\forall P: e_i, e_j \in P} \frac{1}{rank(P)}, i \neq j$$

The score of each entity e_i is calculated as:

$$\vec{f}_{e_i} = \frac{\sum_{j=1}^{E}(f(e_i, e_j) + f(e_j, e_i))}{|E|}, i \neq j$$

We rank the entities based on the above score and select top 100 entities. For every passage in the feedback passages P, we check the existence of the top 100 entities and if the entity exists we add the score the entity with the initial BM25 score. We re-rank the passages based on the new score.

UNH-trema-ecn We start with an entity and passage run. For every query-entity pair, we create an *Entity Context Document*(ECD). To construct this ECD, we filter all passages which mention the entity and "stitch" them together into one "document" about the entity. All entities which occur in this ECD co-occur with the target entity (the entity the ECD is about). We derive a distribution over these co-occurring entities by using the frequency of these entities, that is, the number of times the entity occurs in the ECD. For every passage in an ECD, its score is equal to the sum of the frequency scores of the entities in the passage. We rank the passages using this score. This gives us a passage ranking for every query-entity pair. We call this ranking as a support passage ranking. We obtain a passage ranking from this support passage ranking by marginalizing over the entities.

	MAP	P@R	MRR
UNH-trema-rel	0.07	0.14	0.53
UNH-trema-ecn	0.07	0.14	0.51
UNH-trema-ent	0.08	0.14	0.54

Table 3: Results from TREC CAST

UNH-trema-ent We rank passages for a query-entity pair by the number of relevant entities in the passage. For example, if a passage p contains entities $\{e_1, e_2\}$ and the entities $\{e_1, e_2, e_3, e_4\}$ have been retrieved for the query q, then the score of p for each of the query-entity pairs is $f_{qe_1}(p) = f_{qe_2}(p) = 2$ because the passage has two entities in common with the list retrieved for q. This gives us a passage ranking for every query-entity pair. We call this ranking as a support passage ranking. We obtain a passage ranking from this support passage ranking by marginalizing over the entities.

4.2 Results

5 News

Entity-Ranking Task. In the news track, participants were given news articles and list of referenced entities in articles. The task is to rank the list of referenced entities according to the relevance of each entity to each article.

We submitted 1 automatic run `UNH-Trema-News`.

5.1 UNH-Trema-News

As a pre-processing step for every passage in every article, we first annotate the passages with DBpedia Spotlight and store it in the index. In the method, we first retrieve query-relevant feedback passages using BM25. We create a candidate entity list which consists of all the entities present in the feedback passages. For every entity in the candidate list, we generate an entity-pair, with every other entity in the candidate list. We check the existence of the entity pair i.e. whether both entities of the entity-pair exists in the feedback passage. In simpler terms, we check the co-occurrence of entities in the feedback passages. If the entity-pair co-occurs in a passage, then we propagate the retrieval score of the passage as the entity-pair score.

To calculate the score of each entity, we accumulate the score of every entity-pair and average it with the length of the entity candidate list.

We take the given input list of entities of each query and check the existence of the each input entity in the ranked entities list, if the input entity exists then we take the entity score as the score of the input entity else 0 is assigned as the score. We rank the entities based on these scores. The results are given in **??**.

	MAP	P@R	MRR
UNH-trema-news	0.547	0.424	0.647

Table 4: Results from TREC NEWS - EntityRanking Task

References

[1] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.