CSE474 - Information Retrieval and Extraction
Major Project
Project Description Document (Report)

Team Members
Anuj Goyal (201501129),
Mainak Bhattacharjee (20162101)
Nitin Ramrakhiyani (20172091)
Pratyusha Musunuru (201464090)

Problem: Tail-Query to Head-Query Mapping for Search Engines
One of the biggest challenges for commercial search engines is handling tail queries or queries which occur very infrequently. Frequent queries, also known as head queries, are easier to handle largely because their intents are evidenced by abundant click-through data (query logs). Tail queries have little historical data to rely on, which makes them difficult to be learned by ranking algorithms.

The goal of this project is to develop techniques to map tail queries to head queries, such that the performance of the search engine on such queries can be improved.

Approaches from similar (relevant) problems
There are approaches existing in literature which are focused on learning an IR system by exploring similarity between a query and its relevant documents and also by observing dissimilarity between a query and its non-relevant documents. These approaches become relevant for this problem as the document and query pair can be replaced by the tail query and head query pair and the approaches can then be employed as proposed. A brief about three approaches for document and query pairs can be found below:

1. The first approach is based on the point wise learning to rank technique. It involves an architecture shown in Figure 1. In this approach, a document and query are passed through a feature extraction stage. The feature extraction is completely hand crafted. Then a function learning stage learns a probabilistic classifier to classify the document query pair as related or not. The function learning stage is a machine learning system that can be trained using training pairs of relevant and non-relevant document query pairs and then later tested for an unseen pair. When a test query is tested against the complete set of documents, a rank list of most relevant documents can be derived based on the probabilities obtained from the classifier.

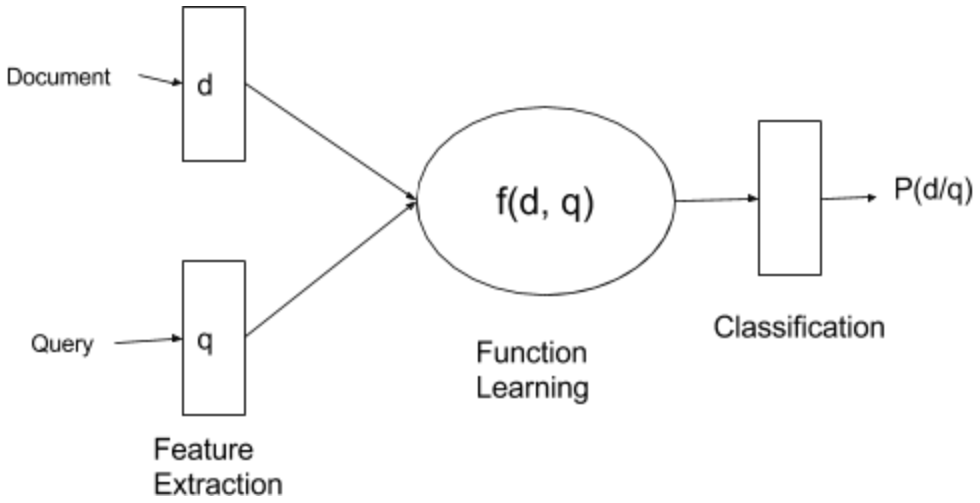

Figure 1: Point Wise Learning to Rank Architecture

2. LambdaMart based Learning to Rank
   In this method instead of traditional learning to rank mechanisms, LambdaMART based ranking is employed. The LambdaMART uses a combination of Decision Trees and Boosting to learn ranks for head queries against a tail query.

3. DSSM
   The Deep Structured Semantic Model or DSSM based approaches can be explored. An important advantage over earlier methods is the use of feature learners given data instances. Feature learners are RNN or CNN based neural networks which can consume textual data (represented by word vectors) and learn salient features from the data without any hand crafting or human intervention. The machine learning classifier then uses these network learned features and performs classification.
   The description on https://www.microsoft.com/en-us/research/project/dssm/ provides multiple references on DSSM and its variations.

Related Work

The following papers are baseline approaches in this area:

1. Verma, Manisha, and Diego Ceccarelli. "Bringing head closer to the tail with entity linking." In Proceedings of the 7th International Workshop on Exploiting Semantic Annotations in Information Retrieval, pp. 37-39. ACM, 2014.
2. Song, Yangqiu, Haixun Wang, Weizhu Chen, and Shusen Wang. "Transfer understanding from head queries to tail queries." In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 1299-1308. ACM, 2014.

As a part of the project, we aim at implementing one of the baselines for reporting and comparison.

Other relevant related work includes:

1. Severyn, Aliaksei, and Alessandro Moschitti. "Learning to rank short text pairs with convolutional deep neural networks." In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 373-382. ACM, 2015.
2. Mitra, Bhaskar, Eric Nalisnick, Nick Craswell, and Rich Caruana. "A dual embedding space model for document ranking." arXiv preprint arXiv:1602.01137 (2016).

Project Contributions

1) Dataset creation

The AOL queries dataset (available at http://jeffhuang.com/search_query_logs.html) is used for creating the query pairs. In this dataset, we have a list of queries followed by user clicked link, rank of the link clicked and timestamp for every query. Queries in the dataset with frequency greater than 150 times are considered as Head queries and similarly, queries with frequency less than 5 are considered as Tail queries. One of the baseline approaches has these figures as 199 and 2 respectively. But on careful observation of the query log we find 150 and 5 as better thresholds.

Heuristic

As the exact pair of head and tail query is not available, a heuristic method needs to be applied to create the pairs. We consider multiple heuristics and choose the one which allows us to form better pairs. The heuristic we follow is - if a particular link is clicked for a tail query and a head query as well, ideally, it means that they have the underlying information need in common partially/completely. This makes them possible candidates for mapping together as positive pairs of tail and head queries.

For a particular user, we take the last clicked link of the query as the link for applying the above heuristic. This is under the assumption that the last clicked link would have satisfied the information need as he did not click any other links beyond that.

Implementation

The above heuristic is implemented through the following steps:

I. We map every tail query T to a set of tuples TL consisting the link and its frequency, which is then sorted descending by the frequency.
II. Next, we map every link *l* that is clicked in tail queries to a set of tuples LH consisting of head queries with their corresponding frequencies. Each tuple is also sorted descending by frequency.
III. Then for every tail query T, we take the link *l* that has the highest frequency, and then go to the corresponding head query set LH mapped to this link *l*. We then link the tail query T with the head query H in LH with the highest frequency.

IV. The process is continued until the tail queries and sets of links and head queries are exhausted. However, a condition is checked while confirming the mapping. A link or head query is considered for mapping only if the
    - frequency fall between link at position i+1 and position i is less than 30% and a link is clicked at least twice for a tail query.
    - frequency fall between head query at position i+1 and position i is less than 20%.

On using the above heuristic on the AOL dataset, we got 130893 positive pairs (that are correct tail-head query mapping). Through an obvious process of linking a tail query to abundantly available non-related head queries we can derive the negative pairs (incorrect tail-head mappings). We also construct such 9637110 negative pairs.

2) Implementation of the baseline paper, *Bringing head closer to the tail with entity linking* by Verma et al.

This paper describes a entity linking based approach to map tail queries to head queries. The authors use an entity linking tool - Dexter to discover the set of entities from a query. Dexter identifies chunks in the query (spots) and links them to corresponding Wikipedia Titles (entities) associating a link probability. To solve the problem of mapping, entities of a tail query *t* and each head query h_i are identified using Dexter and a Jaccard similarity is computed between the two entity sets to score each h_i for the current tail query t. Top k head queries based on this entity linking score obtained as a rank list for a given tail query.

3) THMapNet - A Neural Model for establishing tail and head query mapping

The dataset created provides us with pairs of relevant tail and head queries (referred to as positive pairs) and pairs of non-relevant tail and head queries (referred to as negative pairs). Based on the DSSM idea of feature learning through neural networks, we propose THMapNet, a set of deep neural network architectures which learn relevance between tail queries and corresponding head queries.

THMapNet_v1

THMapNet_v1 has a graph like model instead of the sequential model and allows for two different inputs and a single output. The THMapNet_v1 architecture is described in Figure 2.

Details of the model's layers are explained as follows:

1) Input Layer: The input layer restricts the tail query to contain 10 words each represented by a d dimensional embedding. Similarly the head query is also restricted to contain 5 words represented by the same d dimensional word embedding. This restriction is important for learning a sequence based representation of the input

2) LSTM Layer: The Input layer feeds into the LSTM layer which has two LSTMs one each for the tail and head query. Each LSTM has a hidden layer size of 100 nodes. The LSTM

learns the desired sequence based representation of the queries capturing both semantics through word embeddings and sequence through the time stepped learning.

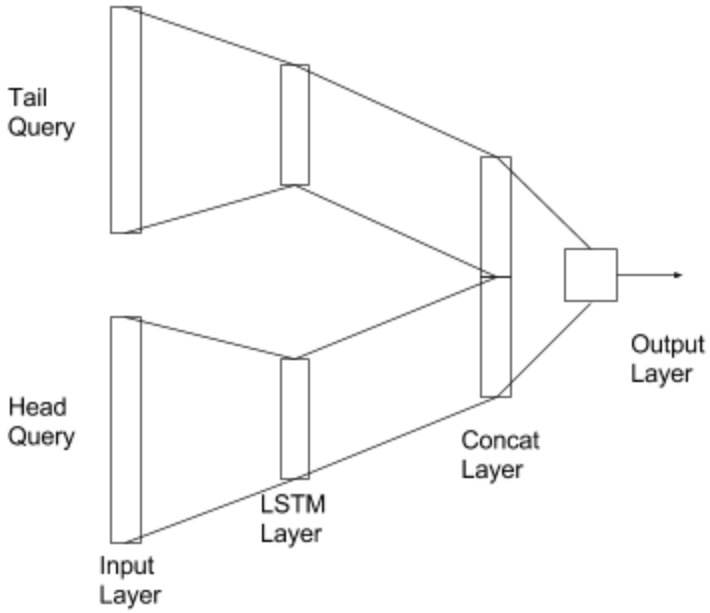

Figure 2: THMapNet_v1 architecture

3) Concat Layer: The 100 dimensional output from the tail query LSTM is concatenated with the head query LSTM's output leading to a 200 dimensional concatenation layer.

4) A dense connection is then made to the output layer with one neuron and sigmoid activation. The single neuron emitts a score which when greater than 0.5 can be considered to be class 1 and 0 otherwise.

Experimentation - THMapNet_v1

As part of the experiments, we employ the 200 dimensional word embeddings from the paper by Mitra et al. mentioned above in the Related Work section. Here we report results averaged over five runs on datasets of sizes 100K and 300K pairs. We experiment with datasets of sizes 100K and 300K pairs containing roughly one - third data corresponding to positive pairs and the rest two-third corresponding to negative pairs. Further we create a test-train split of 80:20 and report the averaged results over five runs.

Data Size	Epochs	Macro F1	Positive Class F1	Negative Class F1
100000	5	82.65	74.74	90.03
	10	82.29	75.25	89.19

300000	5	84.01	78.3	89.4
	10			

THMapNet_v2

THMapNet_v2 has a graph like model different from THMapNet_v1 and allows for multiple inputs corresponding to the tail and head query and a single output.
The THMapNet_v2 architecture is described in Figure 3.

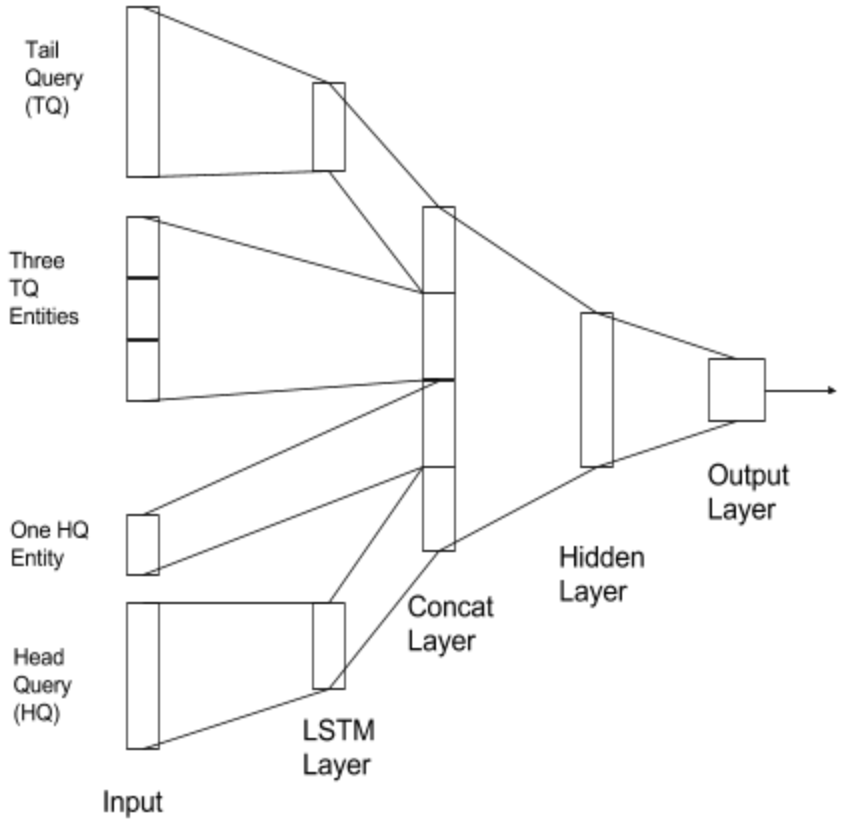

Figure 3: THMapNet_v2 architecture

Details of the model's layers are explained as follows:

1) Input Layer: The input layer restricts the tail query to contain 10 words each represented by a d dimensional embedding. Similarly the head query is also restricted to contain 5 words represented by the same d dimensional word embedding. This restriction is important for learning a sequence based representation of the input. Apart from the word embeddings of the query words, DBPEDIA embeddings of entities in the query are also fed as input. The number of entities for the tail queries is set to 3 ordered by their similarity to the query mention. Similarly one entity is chosen for the head query. The entities are obtained from the Dexter engine as explained in Project Contribution point (2) and have DBPEDIA embeddings of 500 dimensions.

2) LSTM Layer: The Input layers corresponding to embeddings of query words (excluding the entity vectors) feeds into the LSTM layer which has two LSTMs one each for the tail and head query. Each LSTM has a hidden layer size of 100 nodes. The LSTM learns the desired sequence based representation of the queries capturing both semantics through word embeddings and sequence through the time stepped learning.

3) Concat Layer: The 100 dimensional output from the tail query LSTM is concatenated with the embeddings of the three tail query entities as first part of the concat layer of size 1600. Similarly the head query LSTM's output is concatenated with the embedding of the head query entity as second part of the concat layer of size 600, making the total size as 2200 nodes.

4) Dense Hidden Layer: The hidden layer gets as input the concat layer. It is of size 600 (one-fourth the size of the concat layer size).

5) Output Layer: A dense connection is then made to the output layer with one neuron and sigmoid activation. The single neuron emits a score which when greater than 0.5 can be considered to be class 1 and 0 otherwise.

Experimentation - THMapNet_v2
The neural network was implemented using the Keras package in python. But during training of the network we observed that the loss computed at each epoch of training turned either into 'nan' i.e. not-a-number character or negative infinity (-inf) which highlights either problem with the neural network architecture or incorrect use of activation functions. We present this as a negative result of this study. It however calls for a more careful development of such entity and sequence information neural networks.

4) User Study

As the dataset created is purely on the basis of heuristics, it may considered as pseudo-gold benchmark rather than a true gold standard. Hence, it is important to validate the performance of our approach and baseline approaches through human evaluation. For this purpose, we propose to conduct a user study which would help us evaluate performance of systems used for tail to head query mapping. Also for the user study we compare one of the baselines we implemented (as described in Point 2 above) with the THMapNet_v1 architecture.

Design of the User Study
We used Google forms of the format shown in Figure 4, for the user study.
The first section of the form consists of tail query followed by the links when it's fired on google, which helps the user in understanding the information need or context of the query.
The second part consists of the best head query mapped using the deep learning model, and the results when fired on google.

Similarly, the third part consists of the head query mapped by the baseline entity linking system and the corresponding results.
We use a 3-point scale to rate the results generated by both the systems. 0 indicates completely irrelevance, 1 indicates average relevance and 2 completely satisfies the information need.

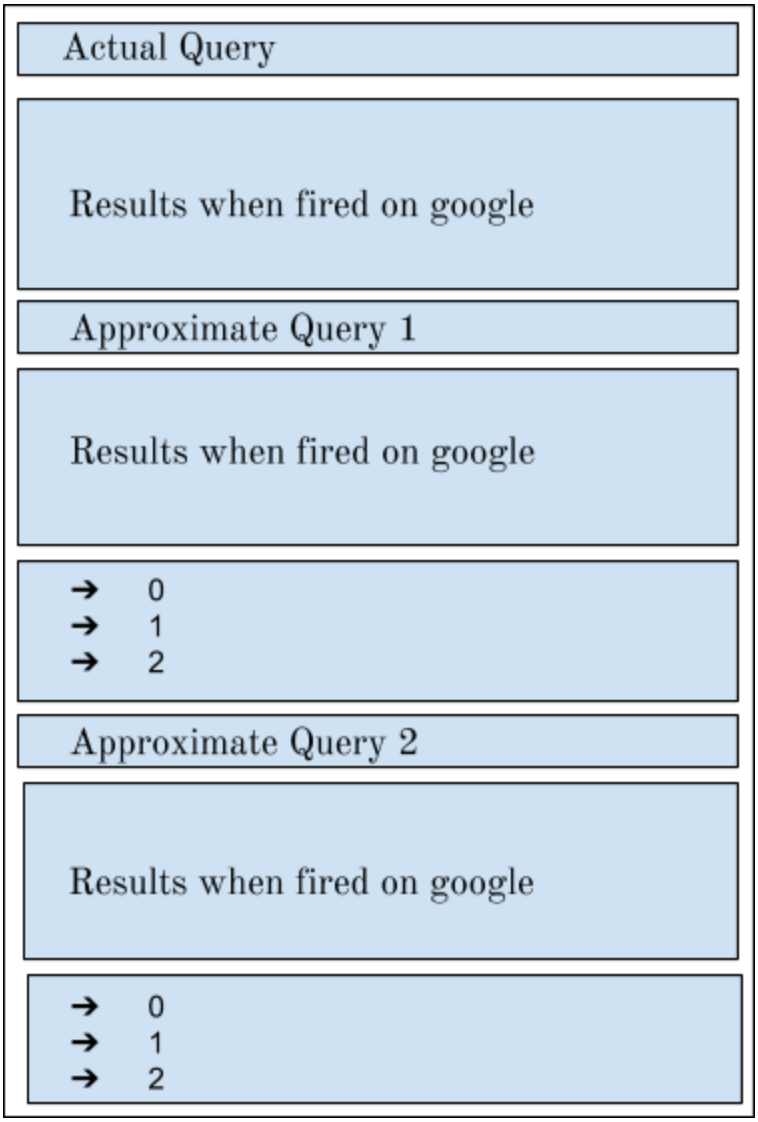

Figure 4: Format of the Google Form used for the User Study

Statistical Inference
We used the T-Test to determine if one of the techniques was (statistically) significantly better in performance than another. We took 30 queries as the initial sample set, since the minimum number of samples required for a T-Test assuming an underlying normal distribution is 30. We made 3 google forms and each with 10 queries. Each form is filled by 10 users and average of the all the 10 ratings is considered for each query. So, we employed 30 users in total. The mean

of THMapNet_v1 system stands at 0.8484848485 and that of entity linking system stands at 0.7545454546. The maximum p-value required for the hypothesis to be true is 0.05 and we got p-value as 0.20 for 30 samples. Through this we can infer that the better performance of the THMapNet_v1 system's is not statistically significant as compared to the entity linking method.

We additionally calculate the minimum sample size required for achieving statistical significance lowering the p-value to below 0.05. Based on the test provided at https://develve.net/t-test.html, the sample size came out to be approximately 550, which is rather infeasible in the current scope of the project.

The google forms used are
https://docs.google.com/forms/d/e/1FAIpQLSfuBumleztA6E2VPjXLGbheqb2-RHZlTQAnEopANNQUCnfeog/viewform
https://docs.google.com/forms/d/e/1FAIpQLScZTYCpWvckzx3ha429n2H6o53SjZplViNRVJOn8wfUEozaKg/viewform
https://docs.google.com/forms/d/e/1FAIpQLSe_f7Xf0GT4lPTrzT3ei0rtoXiyDbRhYPPyips8SqRCfGieHA/viewform

Results of the Statistical analysis
https://docs.google.com/spreadsheets/d/1nvsgpJj0yXv6Lwy6T69ho3uQnkMrp4RVe_1fqHMi1Ss/edit?usp=sharing

References

https://www.ismll.uni-hildesheim.de/lehre/ml-07w/skript/ml-1up-07-evaluation-testing.pdf
https://www.researchgate.net/post/Minimum_sample_size_for_t-test
http://math.ucdenver.edu/~ssantori/MATH2830SP13/Math2830-Chapter-08.pdf
http://www-personal.umd.umich.edu/~acfoos/Courses/381/09%20-%20Hypothesis%20Testing%20with%20t%20Tests.pdf