Course Project for CU Boulder CSCI 5352

Using the Wikipedia Game to Play Trivia Games

Pedro Rodriguez
University of Colorado at Boulder
Boulder, Colorado
p.rodriguez@colorado.edu

Fenfei Guo
University of Colorado at Boulder
Boulder, Colorado
fenfei.guo@colorado.edu

ABSTRACT

A common theme across different factoid question answering methods is to by some means compute the semantic similarity between questions and potential answers and then select the answer with the highest similarity to the question text. A common way to compute this score is to embed the questions and answers in either traditional or neural bag of words vector space, and then rank all possible answers with a machine learned model. In this work we introduce network based methods for using the Wikipedia hyperlink graph to compute an initial list of candidate answers, network-based ranking metrics for ordering these candidates, and text based word embedding methods that are also shown to be effective for ranking using only the page title and question text.

CCS Concepts

•**Information systems** → **Question answering;** Language models; •**Computing methodologies** → *Natural language processing; Supervised learning by classification; Reinforcement learning;* Neural networks;

Keywords

Factoid question answering; network science

1. INTRODUCTION

Question answering (QA) is a subfield of computer science concerned with creating systems that can answer questions posed by humans in natural language. Factoid question answering (FQA) focuses on a subset of QA where questions have distinct fact-like answers. For example, "Who invented general relativity?" which is answered by "Albert Einstein" is an example of (FQA). FQA systems have traditionally incorporated methods from information retrieval, information extraction, and natural language processing.

This work focuses on FQA over paragraphs of text which is the primary task in two trivia games called Jeopardy! and Quiz Bowl. In these games a long paragraph of text is read about an entity and players attempt to guess the answer before the other player. Figure 1 shows a sample question about Albert Einstein from Quiz Bowl.

With Leo Szilard, he invented a doubly-eponymous refrigerator with no moving parts. He did not take interaction with neighbors into account when formulating his theory of heat capacity, so Debye adjusted the theory for low temperatures. His summation convention automatically sums repeated indices in tensor products. His name is attached to the A and B coefficients for spontaneous and stimulated emission, the subject of one of his multiple groundbreaking 1905 papers. He further developed the model of statistics sent to him by Bose to describe particles with integer spin. For 10 points, who is this German physicist best known for formulating the special and general theories of relativity?

Figure 1: Quiz Bowl question about Albert Einstein

One set of techniques in FQA can be seen as embedding the input question in some higher dimensional where questions with the same or similar answers are nearby in the space. At test time, an embedding for the input question is obtained and the returned answer corresponds to some function of the answers of nearby questions (for example the answer of the nearest question). One naive NLP approach would be to embed the text in a higher dimensional space using n-gram bag of words and TF-IDF. Another technique is to use neural bag of words in which dense vector representations for each word in the paragraph are obtained with a skip-gram model such as word2vec, and then averaged together. These methods often also use some form of machine learning and increasingly deep learning to better embed questions with the same answer near each other. All these methods formalize the intuition that the semantic content of questions should be near the semantic meanings of answers.

In Section 3 we describe a novel networked based method for FQA that is loosely inspired by the Wiki Game[3]. In the game players first choose starting and ending pages on Wikipedia, then attempt to find the shortest hyperlink path from the start to the end. The intuition that related pages should be closer on the link graph is used to devise a method for approximating semantic similarity between questions and answers. In this section we show that using network structure alone, specifically excluding textual features, can generate an effective list of candidate answers, and that certain network features are predictive of the true answer.

Section 4 describes and compares text based methods which use only list of candidate answers as input from the

KDD 2017

DOI: 10.475/123_4

network. We specifically exclude the network features derived in Section 3 to observe the effectiveness of text by itself. We also provide additional motivation for using a network based list of candidate answers as opposed to a more traditional multi-class classification scheme.

We then move on in Section 5 to experiments which combine the methods from Section 3 and 4. These results are then loosely compared to the results from a state-of-the-art question answering system for Quiz Bowl called QANTA[7]. We then conclude in Section 6 by detailing ideas for future work.

2. QUIZ BOWL AND WIKIPEDIA DATA

The Wikipedia graph is well suited as the basis for creating a network method since it contains references to a vast number of entities, and encodes rich semantic information in its link graph. However, its also possible that the semantic information is difficult to extract due to the noisy nature of the link graph. Fortunately, prior work has established that crowd-sourced play of the Wikipedia Game can be used to smooth over this noise and compute a semantic similarity distances between pairs of entities[12] that outperforms latent semantic analysis. We chose not to use an explicit semantic network such as WordNet, because although it is much less noisy it does not include less common entities such as Peter Debye in figure 1[9]. Although FQA from semantic networks is not a new idea, we are not aware of any other work which uses the Wikipedia link graph as a semantic network for FQA. We specifically focus on using the Wikipedia link graph for generating answers to Quiz Bowl, a popular academic trivia game[4].

In this section we first discuss the Quiz Bowl task, the subset of questions used, identify important evaluation metrics, and specify how the link graph was built.

2.1 Quiz Bowl Task and Dataset

As briefly described in Section 1 the essential task is to predict the answer to a question given a string of text as soon as possible. In all our experiments we only consider guessing at the end of the question when all the text has been revealed. To evaluate our methods we use a subset of an openly available Quiz Bowl question dataset and compare against the DAN method[6][7]. The dataset contains a total of 36,448 questions with non-empty answers that have a set of 11,895 distinct answers. As will be discussed in Section 4, deep learning methods based on multi-class classification over a closed answer set are only capable of answering roughly half of those questions with a set of approximately 2,000 distinct answers. This limitation primarily arises from the need for a certain minimum number of training examples per answer class. To prove that our method is effective over precisely the set of questions that the DAN system is implicitly incapable of answering, our experiments use a set of 5,624 questions whose answers appear exactly once between all folds of the data.

Lastly, all our experimental results focus on maximizing answer recall $R_a(q, n)$. Answer recall $R_a(q, n) = 1$ when the answer to question q is contained within the first n guesses and zero otherwise. This is emphasized since while this method and the deep learning methods are responsible for generating guesses in QANTA, other components of the system are responsible for reranking candidate guesses based on additional feature extraction techniques such as a language model. It is therefore more important to maximize $R_a(q, n)$ within approximately the top 100 guesses than to maximize precision.

2.2 Building the Wikipedia Link Graph

We now formalize the notion of the Wikipedia link graph, the data collection method, and describe the filtering methods used to avoid nonsensical pages from entering the graph. Our experiments use a pre-extracted adjacency list formatted link graph which includes 5,716,808 pages and 130,160,392 links[2][1]. Let $G = (V, E)$ be a directed graph with vertexes V and edges E. All pages on the Wikipedia link graph correspond to a vertex in V. The existence of a hyperlink from page a to page b on Wikipedia denotes that the edge $e_{ab} \in E$.

In all experiments we filter Wikipedia pages by the content of their title. Specifically, page titles must satisfy the regular expression in `^[a-zA-Z_']+\$`, and not be a stop word[2]. This filtering reduces the graph size to 3,667,749 vertexes and 54,215,111 edges.

3. WIKIPEDIA NETWORK METHOD

With the Quiz Bowl dataset and Wikipedia graph now described in detail we move on to describing the network part of our method. In this section we discuss how to map questions to Wikipedia, identify candidate answers from the mapping, then use network features to rank the candidates. We also discuss ways to make the core shortest path algorithms speed and memory efficient.

3.1 Mapping Questions to Wikipedia

Questions are matched to Wikipedia using exact n-gram matching, and excluding stop words. For example, exact 1-gram matching to Wikipedia pages would result in the yellow highlighted text of the following passage being matched: "With Leo Szilard, he invented a doubly-eponymous refrigerator with no moving parts". These vertexes are considered to be the seed vertex set $\mathcal{S}$ for this question. In the network feature experiments exact 1-gram matching is used, and the text experiments in Section 4 expands to 5-grams. As is readily obvious, there is a great deal of improvement that could be done at this step, some of which is discussed in Section 4 and the rest in Section 6.

3.2 Identifying Candidate Answers

Once a seed set $\mathcal{S}$ of vertexes is chosen then a set of candidate answers is selected. The simplest approach is to consider all nodes within d distance away from any vertex in $\mathcal{S}$ to be a member of a candidate answer set $\mathcal{C}$. Since the Wikipedia graph exhibits small world behavior and every page is connected to every other page by 4.6 links on average it turns out to be sufficient and computationally desirable to limit $1 \leq d \leq 2$. Intuitively, the higher d becomes the less semantically related the nodes in $\mathcal{C}$ are to any of the nodes in $\mathcal{S}$ so the cutoff is reasonable from that perspective as well. For the particular single sentence examined previously, a

[1]This page is now only available by the Wayback Machine, and uses a copy of Wikipedia circa 2009. We plan on collecting an updated version of Wikipedia from the raw dumps soon.

[2]This excludes too much and we plan on allowing numbers and parenthesis in future experiments. Stop words are from nltk.corpus.stopwords

sample of the 150 generated candidates is shown in figure 2 with the source of each candidate identified. When the full paragraph text is used each question has approximately 3,000 candidate answers in $\mathcal{C}$ and 50 seeds in $\mathcal{S}$.

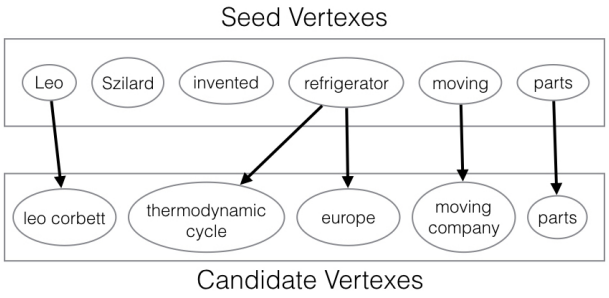

Figure 2: Seed vertexes for the Leo sentence on top with a sample of the 150 generated candidates below. Note how 2-grams would capture "Leo Szilard" who is directly connected to Einstein and that with basic NLP "parts" and "part" could be merged.

3.3 Network Features

Once a candidate set $\mathcal{C}$ is selected the next step is to order the set from most to least likely to be the answer. In this section we focus on describing the network features which were predictive of the answer.

3.3.1 Shortest Paths

As mentioned earlier, various NLP approaches are essentially asking the question of how close each of the candidate answers are to the question text and picking the nearest one. The natural network analog of this idea is to compare the shortest path distances between each candidate and the seed vertexes. Since every sentence and fact in the question relates to a single distinct answer, it is sensible to believe that the candidate nearest in path distance is more likely to be the answer.

Since Wikipedia is a directed graph the distance from the seed to candidates and vice versa is not necessarily the same. Rather than lose potential information, shortest paths are computed both ways.

This then leaves the question of how these list of distances should be converted to numerical scores. To decide that, first note that if there is no path between two nodes their distance they could be treating as being infinitely far apart. The natural solution to this problem is to sum the inverse of the distances which produces scores where higher is better. This is a well known network science metric called harmonic centrality. This is done for both seed to candidate and candidate to seed shortest paths on the graph.

The predictiveness of this is covered in detail in Section 5, but to statistically show why it works we can look at harmonic distance distributions. On the set of 108 questions used in experiments we examined the distribution of distances from the candidates to the seeds. First we establish using figure 3 that these distances follow a normal distribution, and that distances are not heavily skewed. The overall mean distance from any candidate to a seed was 14.4 with a standard deviation of 3.35. For all candidates that turned out to be answers the mean distance increases to 17.34. Since this is in the area of one standard deviation above the distribution it indicates that harmonic distance could be a good predictor of a candidate being the answer.

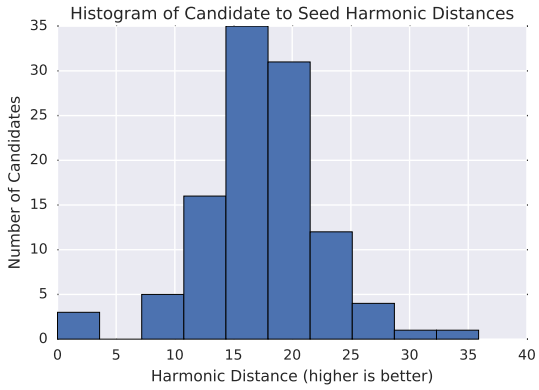

Figure 3: Histogram of harmonic shortest path distance from candidates to seed vertexes. Higher is more indicative of being the answer.

We also tried other ways of aggregating the distance scores such as min, max, and mean distances, but have not seen as much success in the short amount of time used in exploring these.

3.3.2 Degree

Shortest paths do an effective job of picking out candidates, but there is much in the network structure of Wikipedia that can bias results. For example, consider that the distance to highly central nodes like the United States is likely to be very small. Just by being a high degree node the distances to it and from it will be smaller. This is similar to the notion in NLP that some words have a stronger prior likelihood, but that the posterior can be de-biased.

In our network method we use the in and out degrees of nodes to act as this de-biasing mechanism. To handle very high degree nodes we use the logarithm of the in and out degrees as the features for the network classifier.

3.4 Network Feature Classifier

At the start of this project we began with using only the shortest path score which made needing a machine learned model unnecessary. After looking at early errors though it became obvious we wanted to include other features, and a machine learned model is the most effective way to join these together.

In our experiments we use a simple Logistic Regression model that is set to return true if vertex under consideration is the answer and false otherwise. Through cross validation we found that Logistic Regression with L2 regularization strength of $C = .1$ using the SKLearn worked the best. We also used two forms of feature standardization. For shortest path metrics we standardized relative to all data points in the same question, and for degree features with respect to all data points.

Finally, we find that it is important to point out that the classifier is dealing with a large class imbalance problem. Since every question has approximately 3,000 candidates, it is implied that there is a roughly 3,000 to 1 class imbalance. We found that instead of using all of the negative class training examples that we could use random majority class

under-sampling and achieve comparable accuracy with much less volume of data and training time.

To dig deeper on these features we ran a model which did not use quadratic features then obtained the weights. Figure 4 shows the short name of each feature along with the logistic regression weight. Since all features were standardized this is an effective way to determine the effect and strength of each features.

Immediately its not surprising that the harmonic distance sum is a positive predictor (recall that higher is better on this metric). The story is less clear on the other aggregate metrics since they all seem to have the same value regardless of search direction. In looking at the data it seemed that there were large numbers of repeated values that could be making these features not perform well. Due to this these features may have simply become essentially constant factors. Lastly, as expected the degree features act as de-biasing mechanisms. That is, since their weights are negative they negate the inherent centrality of high degree nodes on any shortest path metric.

CS Harmonic Sum	0.446
SC Harmonic Sum	.506
CS Inverse Distance Max	-0.428
SC Inverse Distance Max	-0.428
CS Inverse Distance Min	-0.155
SC Inverse Distance Min	-0.155
CS Inverse Distance Mean	0.446
SC Inverse Distance Mean	0.446
In Degree	-0.0512
Out Degree	-0.104
Log In Degree	-0.188
Log Out Degree	.-0.335

Figure 4: Logistic regression features weights for each network feature. SC indicates seed to candidate distance, and CS indicates candidate to seed distance

3.5 Efficient Many-to-Many Shortest Paths

As previously mentioned the core algorithm used to calculate shortest paths is all pairs shortest paths between the seed vertexes $\mathcal{S}$ and the candidate answers $\mathcal{C}$. Since $|\mathcal{S}| \approx 50$ and $|\mathcal{C}| \approx 3000$ then approximately $150,000$ shortest paths need to be computer per question. In the experimental results discussed in Section 5 we were severely limited in the number of questions we were able to use. The computation time of 100 questions exceeded 1 day and consumed 30GB of RAM which motivated implementing an efficient version of this search. Here we discuss the relevant details of this implementation which we unfortunately ran out of time to use in generating more data for experiments. Our in-progress Rust implementation can be found at https://github.com/Pinafore/qb/tree/guesser-refactor/wiki_network. The current version uses 15x less memory and appears to be much faster, but we are in the process verifying its correctness before using it.

3.5.1 Bidirectional Breadth First Search

Wikipedia is a small world graph which means that as one performs a breadth first search traversal from any node the number of nodes encountered is large. Another way this could be seen is that the branching factor b of the Wikipedia graph is large. This motivates the use of bidirectional breadth first search. The method is relatively straightforward in that the algorithm alternates between advancing a forward looking BFS on the regular adjacency list, and a backward looking BFS on the reverse adjacency list. At some point these searches meet in the middle.

We were able to experimentally verify that the mean shortest path lengths of the paths found in our data agrees with the $d \approx 4.6$. Figure 5 shows a histogram of all shortest path distances in our experimental data. If one knows that the mean distance d between knows is relatively fixed, then the algorithmic efficiency of ordinary BFS can be written as $O(b^d)$. Bidirectional BFS changes this performance bound to a sum of searches with complexity $O(b^{d/2})$. Theoretically and in practice this yields a significant boost in performance.

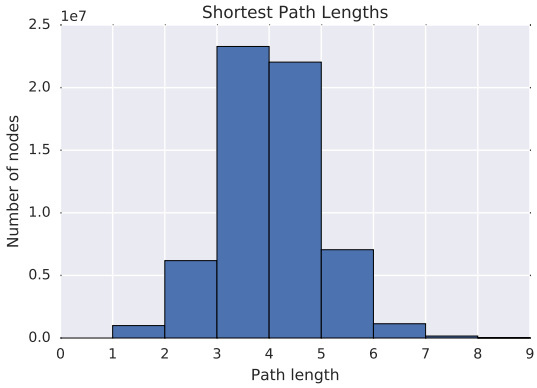

Figure 5: Histogram of shortest paths for shortest path length pairs between $\mathcal{S}$ and $\mathcal{C}$ in both directions for all questions.

3.5.2 Early Stopping

Figure 5 also shows that it is unnecessary to run shortest paths to completion since the vast majority of nodes do not have path length greater than 6. We therefore wrote an early stopping rule into our algorithm which will terminate the BFS once it reaches a maximum path length of 6.

3.5.3 Caching

The final optimization made in our many-to-many shortest path algorithm is caching. This optimization becomes obvious if one sees the search as a double for loop wrapped around a single source, single destination shortest path search as seen in figure 6. In this case it would be desirable to cache the distances found in the outer loop when iterating over the inner loop. In this example it would mean caching the seed distances while iterating over the candidates which would decrease the number of times those distances would otherwise be computed by a factor of approximately 3,000 at only the expense of keeping the seed distances for a particular seed saved in memory.

4. TEXT FEATURES

Until now we have described the network only methods used in our method. This was done specifically so that our experiments can measure the effectiveness of the network

Figure 6: Calculate shortest paths between $\mathcal{S}$ and $\mathcal{C}$
Require: Wiki Graph G, seeds $\mathcal{S}$, and candidates $\mathcal{C}$
Ensure: $paths = Map[(seed, candidate), distance]$
  $paths \leftarrow$ empty map
  for all $s \in \mathcal{S}$ **do**
    for all $s \in \mathcal{C}$ **do**
      $paths[(s, c)] \leftarrow shortest\_path(G, s, c)$
    end for
  end for

features, and text features separately. We now discuss the text features we use.

With a finite set of answers as is the case for Quiz Bowl, FQA tasks are often treated as multi-class classification problem over the answer set. This is how the DAN method works, but unfortunately it introduces the problem that sufficient training examples are needed per answer class. Similarly to the network classifier, we instead take an approach to binarize the task. That is, the task is to given a candidate answer and its extracted features, decide whether or not the candidate answer is correct.

In this section the only network method upon which the text features depend on is obtaining an initial list of candidate answers. This allows for easy comparison between the methods, and there isn't an obvious way to generate a smaller list of candidate answers. Its also computationally expensive and introduces noise to for example just consider all possible answers.

Thus, the task is for text features to rank the candidate answers from the candidate answer set $\mathcal{C}$. To accomplish this we make use of unsupervised machine learning to compute a similarity score between the question text and the text of the answer.

There are lots of possible techniques we can explore to design the text features, such as the bag-of-word model, n-gram features, the TF-IDF features and distributed representations of words. In this paper, we choose to use distributed representations of words which are also known as word embeddings that have been trained on the wikipedia corpus. The choice to use word embeddings is primarily due to the rich semantic and synthetic relations that it captures. For example, word2vec famously showed that the model captured the relationship between man, women, queen, and king[8]. It is also possible to train such a model on Wikipedia text directly so that the semantic relationship between Wikipedia entities can also be captured.

Several models could be used for learning a dense word embedding such as matrix factorization based methods such as Singular Value Decomposition, Global Vector Embeddings (Glove)[10], and the word2vec[8]. In this paper, we use the word2vec model to learn word embeddings on a snapshot of Wikipedia from 2014.

4.1 Skip-Gram model

The Skip-Gram model is one of the most popular word2vec models that learns word embeddings by maximizing the probability of the surrounding words given a pivot word in a sentence.

4.1.1 Canonical Skip-Gram Model

The canonical model learns two sets of parameters $X_w = \{v(w) \in \mathbb{R}^d, \forall w \in V\}$ and $X_c = \{v_c(c) \in \mathbb{R}^d, \forall c \in V\}$, where d is the dimension of the vector space to which we project the words, V is the vocabulary, X_w is the word embeddings we expect to learn as final results, and X_c is the context embedding which we can view as the weights in logistic regression models. For a given word pair (w_t, c), the model defines the probability that word c is observed in the context window of pivot word w_t as the softmax over all the vocabulary:

$$P(v_c(c)|v(w_t)) = \frac{exp(v_c(c)^\top v(w_t))}{\sum_{w_i \in V} exp(v_c(c)^\top v(w_i))} \tag{1}$$

To prevent the trivial solution and pushes the unrelated word vectors away from each other, the model introduces negative sampling by randomly sampling some non-observed words from a certain distribution and minimizing the probability $P(v_c(c)|v(w_t))$.

The goal is to maximize the objective function $J(W, C)$ in equation 2.

$$J(W, C) = \frac{1}{|V|} \sum_{w_i \in V} \sum_{c \in Context_w} logP(v_c(c)|v(w_i)) \tag{2}$$

4.1.2 Negative Sampling Skip-Gram model

The computation of the softmax layer in the canonical Skip-Gram model is very expensive, because the normalization factor requires iterating over all the words in the vocabulary V, which can be very large. To address problem, [**?**] proposed two solutions: Hierarchical Softmax, which is a computationally efficient approximation of the full softmax, and the Negative Sampling model.

The negative sampling model is a simplified version of the Negative Contrastive Estimation. [**?**] suggest that it is not necessary to exactly capture the same objective as long as the vector representations retain their quality. They define the negative sampling (NEG) objective as (for one central word w_i, context word pair w_c),

$$\begin{aligned} logP(v_c(c)|v(w_i)) &= \log\, \sigma(v_c(c)^\top v(w_i)) \\ &\quad + \sum_{j=1}^{k} \mathbb{E}_{w_j \sim P_n(w)}[\log\, \sigma(-v_c(w_j)^\top v(w_i))] \end{aligned} \tag{3}$$

where $\sigma(\bullet)$ represents the sigmoid function $\frac{1}{1+exp(-\bullet)}$;

and the negative sampling distribution $P_n(w)$ is proportional to the 3/4 power of word frequency.

$\sigma(v_c(c)^\top v(w_i))$ can be viewed as the probability that w_c is observed in the context of central word w_i, and for negative sample, the $\sigma(-v_c(w_j)^\top v(w_i))$ is the probability that w_j is not observed in the context of central word w_i. Each pair of word forms a logistic regression.

4.2 Ranking score based on BOW

The Skip-Gram model learns two sets of embeddings $X_w = \{v(w) \in \mathbb{R}^d, \forall w \in V\}$ and $X_c = \{v_c(c) \in \mathbb{R}^d, \forall c \in V\}$, each for words and contexts. In order to rank the possible answers in the candidate sets, we compute the ranking score by:

$$P(\theta_a|\phi_q) = \frac{1}{1 + exp(-dot(\theta_a, \phi_q))} \tag{4}$$

where θ_a represents the vector representation of a possible answer, and ϕ_q represents the vector representations of the question. $P(\theta_a|\phi_q)$ estimates the probability of having answer a given the context of question q.

Each possible answer is a list of words $a = [w_a^{(i)}]_{i=1}^m$, for example, the answer *Albert Einstein* contains two words $[Albert, Einstein]$. Each question is also a list of words $q = [w_q^{(j)}]_{j=1}^n$. To have the vector representation of answers and questions, we use the bag-of-word model, i.e., the average of word/context embeddings. In this case, we view the questions as contexts of answers. Therefore,

$$\theta_a = \frac{1}{m}\sum_{i=1}^{m} v(w_a^{(i)}) \tag{5}$$

$$\phi_q = \frac{1}{n}\sum_{j=1}^{n} v_c(w_q^{(j)}) \tag{6}$$

The reason of using word embeddings for answers and context embeddings for questions is that it fits the objective function of the skip-gram models which maximize the co-occurrence probability of a word-context pair (w_i, w_c) .

4.3 Multi-sense word embedding

One disadvantage of the word embeddings is that they overlook the homonymy and polysemy words. Many experiments have proved that learning multi-sense word embeddings will help to improve the performance of a variety of tasks, especially tasks that rely on semantic relations of words. Therefore, we also run experiments with the sense embeddings of words. And we learn the sense embedding by soft-attentional skip-gram model.

5. EXPERIMENTAL RESULTS

The primary evaluation of our methods is to compute the recall R for each question. The main focus is on maximizing recall within the first $n \approx 200$ guesses.

In all the following experiments we use the subset of 5,624 questions which the multi-class classification based neural methods are inherently incapable of answer since there is exactly one instance of the answer.

Additionally, due to computational difficulty the experiments on the network based features were only able to use a set of 210 questions to perform evaluations on. We are actively solving this problem with the efficient many-to-many shortest paths algorithm described in Section 3, but do not have results yet.

5.1 Network Features

To test our experimental setup was as follows. We selected the following set of features to extract from each of the distances from seeds to candidates and candidates to seeds: harmonic distance, maximum inverse distance, minimum inverse distance[3], mean inverse distance, in degree, logarithm of in degree, out degree, and logarithm of out degree. This was further extended by using quadratic interaction between all features. The Logistic Regression model in SKLearn was used to train a model with regularization strength $C = .1$.

[3]Excluding the first entry since it is implicitly 1 since the candidate was by definition $d = 1$ away from at least one seed

Experimental results for this method are shown in figure 7. As noted before, a set of 200 questions was used for evaluation. All of these question's answers exist on the Wikipedia graph, however only 108 of the question's answers are captured using vertexes $d = 1$ away from the seed vertexes as candidates. Thus, the network features based method can at best accomplish a recall of $108/200 = .54$.

To evaluate the binary classifier random 75 train/25 test splits of the question were used. For each split, the model was trained on the training set and recall was computed on the test set. The data from figure 7 was generated using 5 of these random splits. Variation between these are shown via the error bars.

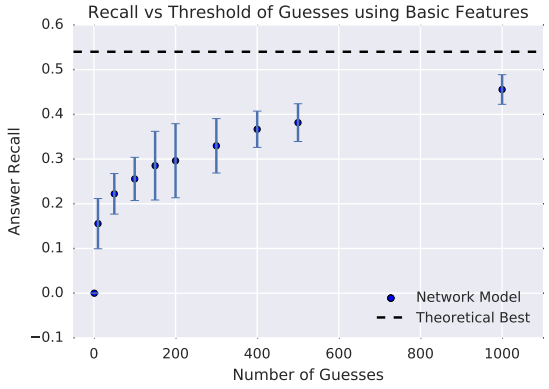

Figure 7: Answer Recall versus the number of guesses used. This plots recall against the 200 question set and shows the maximum attainable recall of the method

5.1.1 *Random Majority Class Under-sampling*

In exploring ways outside of simply getting more data to make our model better we looked into random majority class under-sampling. This method attempts to balance the class ratio between positive examples (true answers) and negative examples (wrong candidate answers). The effect is twofold: the learning problem is easier, and since less data is used it is computationally efficient. Without using this method, the binary classifier trained for figure 7 trains on 256,728 examples.

With majority class under-sampling the total number of training examples goes down to 162[1]. Experimental results using under-sampling are shown in figure 8. Although the model runs significantly faster, figure 8 shows that the overall performance is not too greatly affected. The greatest effect in fact is that the variance increases. For both these reasons this classifier was run for 1- randomized folds rather than the 5 from the original classifier. With an increasing volume of data from a faster network feature extractor majority class under-sampling seems like a promising approach for dealing with the large class imbalance problem.

5.2 Unsupervised Text Features

In Section 3 we describe how we generate the candidate answer set $\mathcal{C}$ by selecting all nodes within d distance away from the seed set $\mathcal{S}$. For $d = 1$ this is simply the the neighbors of the seed vertexes. Since the text methods are currently computationally efficient we also consider run experiments

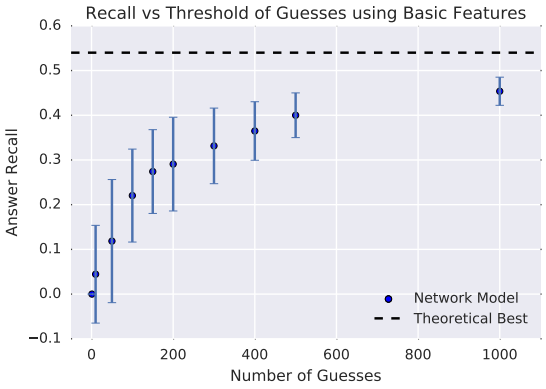

Figure 8: Shows the same data from 7, but using random majority class under-sampling

allowing any nodes within $d = 2$ to be included in $\mathcal{C}$.

To study how the algorithm performance with different hyper-parameters, we sampled 210 questions from the total hard question set, and compared the recall of the algorithm under different settings. Results are show in Figure 9. For this figure, we can make two observations:

1. Generating seed vertices by matching up to 5-grams can largely increase the recall by increasing the upper bound.

2. Enlarging the candidate sets by adding in more vertices that have *geodesic distance* $= 2$ from the seed vertices does help to increase the upper bound to some extent. However, it decreases the recall within top 100 guesses.

Based on these two observations, we can infer the conclusion that including more candidate vertices by enlarging the *geodesic distance* is a bad option, which is also reasonable, because it will include not only several potential answers but also much more negative answers at an exponential order; on the other hand, matching seed vertices with up to 5-grams can help with increase recall as we expected, for the reason that it will not increase much irrelevant answers.

Then we compare the recall of using sense embeddings and word embeddings, and study the true performance of this methods by comparing with the upper bound of the network based candidate-generating model we used under different settings, i.e., the ratios of questions that have answers in the candidate sets. Results are shown in Figure

From Figure 10, we can make several observations and conclusions,

1. We can see that sense embeddings do help to achieve larger recall in top k guesses, where $0 < k < 200$.

2. The word/sense embedding based unsupervised model converge quickly in top 200 guesses, and it can reach relatively near to the upper bound.

3. Almost 80% of answers are include in the page titles of the wiki graph. However, only 58.6% of answers are in the neighbors of the seed vertices, this maybe the bottleneck of this model.

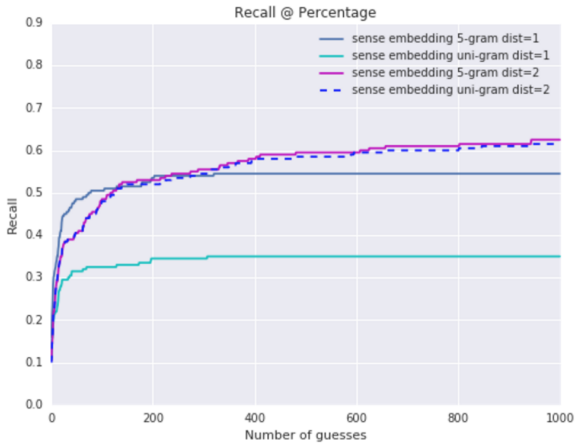

Figure 9: Comparison of the recall of unsupervised methods based on sense embeddings under different hyper-parameters for selecting candidates vertices in wiki graph.

Combining the observation we made from Figure 9 and Figure 10, we can make the conclusion that to increase the recall, we should study how to include more answers in the candidate sets.

One possible solution could be increase the number of the seed vertices and map page tiles to true answers by partial string matching or word similarities instead of exact string matching. Another way is to design better strategies of generating candidate vertices based on network properties by studying the missed answers.

5.3 Network and Text Features Combined

Unfortunately we did not have time to perform experiments that incorporate both of these features. However, as seen in our preliminary experimental results it appears that there is much to be gained in using the Wikipedia network structure even if to simply generate candidate answers for text methods to use. We also note that the network based classifier methods are very data-starved so its performance is expected to improve. For instance, note that with quadratic features the number of features is on the same order as the number of positive training examples. As shown with under-sampling these were by far the most important for training. Even then, the basic network methods shown in figure 7 achieved about 30% recall at 200 guesses. Similar performance at 200 guesses could be seen in the embedding method that was similarly restricted to $d = 1$ and unigram page matching in figure 9 (light blue line). Figure 9 also highlights an obvious area of improvement in extending the network methods to capture using higher order n-grams. We are most interested in performing experiments to find whether these methods are both finding a set of common "easy" answers or are capturing relatively disjoint sets of answer. If the second is the case then overall combined performance could be boosted even further.

6. CONCLUSION

In this work we introduce a novel method for FQA based on using the network structure of the Wikipedia link graph. In both the network and the text ranking based methods

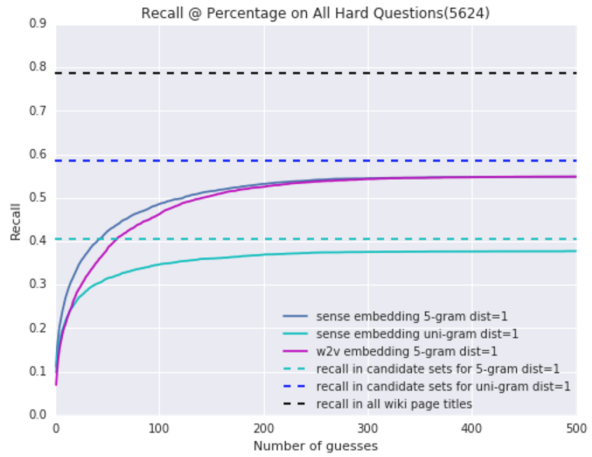

Figure 10: (1) The solid lines compare the recall of using sense embeddings and word embeddings. (2) The dotted lines show the upper bound of the network based candidate-generating model under different settings

discussed in Sections 3 and 4 respectively we show that this is an effective mechanism for narrowing down to a more likely set of answers. We also show that this list of on average about 3,000 vertexes can be further cut down to a list of 200 guesses without decreasing overall recall by too much.

With network features this was done with using shortest path distances from the seed vertexes $\mathcal{S}$ to the candidate vertexes $\mathcal{C}$ and vice versa. Additional features such as in-degree and out-degree were also incorporated.

Text features achieved good recall by using unsupervised word embeddings trained on Wikipedia to compute a semantic similarity score to rank by. Experiments also showed that more permissive methods of allowing vertexes into the seed set $\mathcal{S}$ can be used to increase the maximum recall possible. However, it is still unclear which methods would overall perform better. For example, recall with $d = 1$ was superior for low n in figure 9, it achieved a lower maximum recall than a candidate set of $d = 2$ vertexes.

Experiments have also not yet been performed comparing the effect of each feature independently, and when combined together. It seems that since these methods are quite orthogonal in approach the additive performance effect should be large, but this will need to be confirmed experimentally.

These results are especially encouraging since reasonable performance is being achieved on a set of questions for which DAN and therefore the current QANTA system are incapable of answering. Even then, there still remains ample areas for improvement in both of these methods by increasing recall at a small number of guesses, and increasing the maximum attainable recall.

7. FUTURE WORK

Through discussion of all these methods several directions for future work have been highlighted. In this section we explicitly enumerate the methods we intend on experimenting with.

1. Use the text features in the network classifier to determine how additive the features are in recall performance.
2. Use the efficient many-to-many shortest paths to create a larger training set.
3. In the network method increase 1-gram matching to 5-gram matching.
4. In both the network and text methods test max length n-gram matching. For example, “Leo Szilard” would match “Leo Szilard”, but not “Leo” or “Szilard” since that text would already be matched.
5. In both the network and text methods limit entries in the candidate set to those present in the quiz bowl answer set.
6. Use better Wikipedia entity disambiguation and NLP stemming techniques to produce higher quality page matches[11][5]. For example, in figure 2 “parts” should map to “part”.
7. Perform error analysis over different classes of errors. First, look at examples when the answer was not captured and the feasibility of changing the matching rules to improve this. Second, look at examples where the answer is captured but not recalled to the top 200. Finally, determine why many answers do not match to entities on the Wikipedia graph.

8. REFERENCES

[1] scikit-learn-contrib/imbalanched-learn: Python module to perform under sampling and over sampling with various techniques.

[2] Using the wikipedia link dataset - henry haselgrove. https://web.archive.org/web/20160516004046/http://haselgrove.id.au/wikipedia.htm.

[3] Wikipedia:wiki game - wikipedia. https://en.wikipedia.org/wiki/Wikipedia:Wiki_Game.

[4] J. Boyd-Graber, B. Satinoff, H. He, and H. Daumé III. Besting the quiz master: Crowdsourcing incremental classification games. In *Empirical Methods in Natural Language Processing*, 2012.

[5] X. Cheng and D. Roth. Relational inference for wikification. In *EMNLP*, 2013.

[6] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*, 2014.

[7] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*, 2015.

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

[9] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[10] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[11] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL*, 2011.

[12] R. West, J. Pineau, and D. Precup. Wikispeedia - An Online Game for Inferring Semantic Distances between Concepts. *IJCAI*, 2009.