Question Answering

Muhammad Umer Altaf (778566), Lufan Zhang (827495), Chi Che (823488)

Kaggle Team Name: "UniMelb"

1) Introduction

In this project, we implemented a Question Answering System. The goal of the system was given a Wikipedia article and a list of question based on the article, answer as many of them as possible. As per the project specification document as Baseline QA system was developed which yielded 12.5% accuracy. Then we implemented 3 enhancements, mainly: a better question type classifier, better sentence retrieval engine and a probabilistic QA system based on semantic cues. These enhancements enabled us to achieve maximum accuracy of 13.2%. In this report, we will outline the challenges we faced and the solutions we implemented

2) Basic Question Answering (Base QA) System

The Base QA is constructed as per the requirements in specification document, but we did some changes to few components (mainly sentence retrieval and answer formatting), our system works on following architecture:

- For each question:
    - Retrieve top N most relevant/similar sentences using a TF/IDF model (this is done in blocks, refer Error Analysis part)
    - NER tag the relevant sentences and add the NUMBER and OTHER tag manually, also we change ORGANIZATION to OTHER
    - Classify question into possible answer types ["PERSON", "LOCATION", "NUMBER", "OTHER"] using simple rule based approach (which uses manually defined lists of words for each type, and if words in question match any word in the list, that list is inferred as QuestionType of this question)
    - Combine concurrent same type entities.
    - Now that we have NER tagged relevant sentence, we extract QuestionType entities (E) from it and then apply following filtration rules:
        - If possible answer in (E) is a complete substring of question text we discard it
        - Sort the remaining possible answers as to their distances from the open class words in the question (for open class words, we used a simple approach, which is we removed all the stop words from the question and assumed that remaining words are OpenClass words, this way we did not had to do POS tagging here)
    - Then we select the first answer from the sorted list as guessed answer and clean it a bit using simple string processing to match expected format as observed during error analysis.

Error Analysis of Base QA

After implementing the above system, we spent some time looking at the errors our system was making. Some of these errors are listed below (grouped into categories):

a) <u>Sentence Retrieval:</u>

The tf/idf model was evaluated using the DEV data set, we found that it is 59% accurate (i.e. for how many questions top result by the model matched the key "answer_sent").

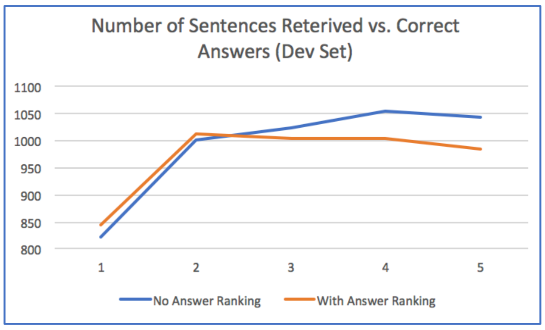
To counter the remaining 41% answers, a strategy was used where we extract not only the top result, but subsequent N top matches too, results are illustrated in plots below. We tried both with answer ranking (the third rule in specs) on and off (if answer ranking is off we just filter the possible answer for required entities and select the first element). These plots offer very interesting insights.

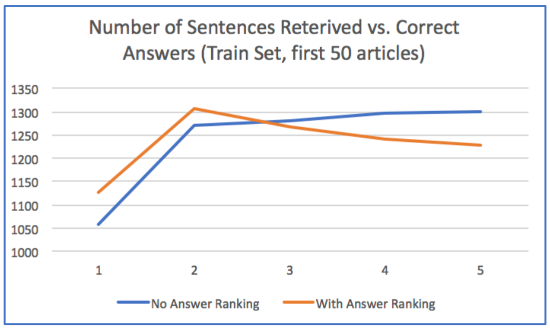
Notice that number of correct answers increase as we retrieve more sentences per questions, reason being that suppose that top sentence does not has the required QuestionType entity, we might be able to find it in next sentence.

Another insight we got was that, if we do answer ranking (i.e. distance from open class words in question with each possible answer), accuracy increases for up to two sentences and decreases after this. The reason we can suggest is this that as more sentences are retrieved, the average number of candidate answers per question also increases, decreasing the probability of selecting the right answer. In fact, we measured this and found that for each number of sentences retrieved the average length of answer list also grew as stated in Table 1.

No of Sentences Retrieved	Average Length of Answer List
1	5.4
2	7.1
3	9.2
4	11.7

Table 1





Implemented solution to sentence retrieval problem:

To address this problem we, did blocked sentence reterival where we combine the top 2 sentences for a question by concatinating them and labeling them as bestSentence, and then concatinating 3rd,4th,5th and 6th sentence labeling them secondBestSentence and while searching for entities we first serach in bestSentence and only if we don't find required entity we search in secondBestSentence. We did this with answer ranking ON. This improved our accuracy on DEV set to 0.1251 compared to 0.0997 with only one sentence fetched.

b) Entity Extraction and Question Classification:

The first problem we noticed was that we were doing a lot of errors in tokenization and preprocessing, for example, answers may contain some punctuation and stop words (, -, The) so we allowed a few of these in our reterived sentences as special rules. Second issue was that Stanford NER was trained on some other kind of data and these wikipedia articles were a bit different, for eaxmple it could not tag newer terms like "Eminem" as PERSON, instead we had to tag it as OTHER. Also sometimes the tagger did not tag complete entity like "The President" only had president tagged and not the "The", but answer expected it. Further, we also ourselves, did some mistakes combining concurrent tokens of same type, for example "George W. Washington" was pretty hard to get as we couldn't be sure that wheather the sentence gets ended after W (i.e. it's a full stop) or its part of bigger entity. We did try a few hard coded rules but were not very successful in addressing this problem.

For NUMBER type we had tried to include dates, percentages and currecny in this class, and it was a bit difficult to tag some numbers completely like "USD$ 5.5 million". Exact numerical answers were a bit easy (in fact we got 70% to 80% of our accuracy only from this class).

Entity extraction can always be improved using more and more special rules by trying to mimic the actual anwer format in the Train Set, but here non uniformity of answer format was apparent, same answer could be given in many formats (10%, ten percent) and wordings ("about 2"), and sometimes these special rules clashed.

Also during analysis we found that we did most errors for the QuestionType "OTHER" and this very type was classified most by our rule based question classifier (3/4 times more likely). We addressed this problem in our first enhancement.

Accuracy of Base QA

We started from about 1.03% correct answers and then tried to address most of the problems mentioned above and ended up with 12.5% accuracy on DEV set.

3) Enhancement 1 (Question Classifier)

As mentioned aboive that we were having hard time with the “OTHER” entity, and it became difficult to distinguish between “PERSON” and “LOCATION” using hard coded rules. So to address this problem we built a suppervised question classifier using following steps.

- Prepeared the training dataset for the classifer from the last 300 atricles of the train set using following method:
    - For each question we NER tag the correct answer sentence (we also manually add NUMBER)
    - Locate the answer in the sentence and get its tag, this is the infered QuestionType
    - Save each question and its infered type in a .csv file
- Train a Stochastic Gradian Descent Linear model uaing this .csv file using following steps:
    - Preprocess the question sentences by tokenizing and lemmatize them to base nouns.
    - Vectorize using a TF/IDF vectorizer and fit a SGDClassifier over the 80% split of the above .csv file, use remaining 20% to judge accuracy of our classfier.
    - Save model to disk
- Load and use this model to classify new questions.

The linear model we trained resulted in about 69% accuracy over the 5 classes [NUMBER,PERSON,LOCATION,OTHER,ORGANIZATION] (Note: we can disregard ORGANIZATION at runtime but its nice to have more classes as to separate other required classes).
Following table compares enhancement accuracy with BaseQA.

Model Used	**Accuracy – Dev Set**	**Accuracy – Train Set (First 50 Articles)**
Base QA	0.125856771445	0.144521646179
Enhancement 1	0.131765540061	0.152218065206

We got about 1.4% increase in accuracy over the Dev set, and saw similar improvement on Kaggle score. This method proves that automatically learning rules is much better than listing manually.
Although this method works well but still it has some flaws, most obvious is this that, to fit classifier we used inferred QuestionType, not the actual real type of the question. This real type is not available and for a pure supervised system they should have been manually added to the dataset.

4) Enhancement 2

For this enhancement model, it follows the common QA approach including question processing, passage retrieval and answer extraction.
As for the question processing, two main tasks are performed. The first one is answer type detection, which exploits semantic question information like part-of-speech tag to predict corresponding answer type. For this task, supervised machine learning method is used to train a classifier to predict the answer type of the test set question. The other task is query formulation where the keywords of the question are extracted to form a query to feed into the IR search engine. In this approach, we simply leave out the wh-words in the questions and the rest of the words are used as query terms.
For the passage retrieval, we simply use IR techniques to find relevant sentences and then based on the predicted answer type we filter out those sentences which are not likely containing the answer type.
In the answer extraction section, to extract the answer term exactly matching the correct answer. We have tried several approaches. Initially, as a basic assumption that factoid answer will not contain same words as in the question, thus we filter out the repeated words occurring in both question and sentence candidates. Then we make attempt to return the first NN or NNP word matching the question focus as the answer, but we recognise that most of correct answers consist of multiple words. Then we try to get ngram of the sentence candidate and return the ngram with the most NN or NNP terms as the answer. Unfortunately, the size of correct answer varies from one another and it is not accurate to specify the number of words to be returned. By observing the error, it is obvious that most of time we can get the correct sentence candidates, however, the problem arises when we choose which words we return as the correct answer. Therefore, we come up with an idea that we can make use of the same words

appearing in both question and sentence to get the index span of the sentence, where the answer should outside the range of the span index. Next, we can get the answer with various size by applying POS tag to the rest of words in the sentence. Finally, the answer with the highest scores are returned by applying several rules to limit the choices of candidates.

5) Enhancement 3

Most of the questions are classified as OTHER, in which many questions start with 'what'. For this kind of questions, my enhancement will find the first NN word in question and get its first Synset of noun. After that, in every given POS tagged answer text, the program will find all the candidate answers of whose Synsets at least one Synset meets one of three conditions (Synset is equal to question Synset or Synset is hyponymy of question Synset or Synset is meronymy of question Synset.
In this way, more relevant candidate answers will be returned.

Model Used	**Accuracy – Dev Set**
Base QA	0.12585677144883006
Enhancement 3	0.12621129756558733

6) A Better Evaluation Metric

The default evaluation metric (exact gold string matching) for this project is very harsh, especially considering the quality of the dataset (i.e. non-uniform answer formats), complexity of the task and amount of information available. It would be nice to allow a little flexibility while evaluating, we have proposed and implemented following metric (available in Enhanced QA and can be turned on or off):

- If proposed answer is a substring of actual answer or the actual answer is the substring of proposed answer, we award 0.75 mark, we added a check that we don't award points for only stop words.
- If we are suggesting a list of possible answers and actual answer happens to be in that list we award 1/(position of actual answer in proposed list), for example if actual answer was the second answer in the list we award 0.5 mark.

This metric is by no means perfect and can be improved to consider the length of the matched segment of the actual answer in the proposed answer. Also, the marks mentioned above (0.75, 0.5) can be tuned by careful examination of data (this will be very tedious). In the following table, we compare the evaluation results of the Enhanced QA using original metric and proposed new one, over 2 datasets.

Metric Used	**Accuracy – Dev Set**	**Accuracy – Train (first 50 Articles)**
Exact String Match	0.1317655400	0.152218065206
New Metric	0.4077936658	0.405692143239

This shows that our Enhanced system is not so bad after all, we can answer unto 3 to 4 times more answers correctly if we are given a little flexibility over the answer format and answer attempts.

7) Improvements

Our system heavily focused on text retrieval strategies and these models can only get you to a certain level. For a better QA system, some kind of semantic analysis of the data is necessary. Due to the lack of time we were not able to explore these methods, but knowledgebase generation via factual rule learning seems very interesting.

Then there is a possibility of using better pre-trained models like 7 class NER tagger [1], which may be able to tag NUMBERS better. Also, we can use better word embedding (tailored to the task of QA) as demonstrated by [2], these word embeddings can help us build a better (CBOW) Contextual Bag of Words representation. Other things can be tried like better question modelling, via query expansion techniques [3], which helps understand more about the meaning of the question.

8) References

- [1] J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370. http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf
- [2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.
- [3] X. Li and D. Roth, Learning Question Classifiers: The Role of Semantic Information Journal of Natural Language Engineering (2005)