1. Summary

The solution is based on a Logistic Regression (LR) 3-class classification model using Information Retrieval (IR), neural network embeddings (W2V) and heuristic/statistical corpus features. I spent the first weeks of the competition trying to build a relevant knowledge base (KB) and understanding how the features from the questions and the answers could be combined in the most optimal way. The last two weeks were mostly dedicated to model tuning and feature selection. The rationale behind using 3 classes (Answer A, Answer B, Other) instead of 4 (one per answer) was to augment the training dataset and deal better with low confidence predictions. I've found that questions with short answers (less than 4 words) were positively sensitive to n-gram matching (more than 0.7 accuracy by using IR alone) while the longer ones required combined knowledge (inference) to be solved correctly. My tools of choice were Elasticsearch 2.0 (ES), Gensim and the Sklearn/Numpy stack.

2. Features Selection / Extraction

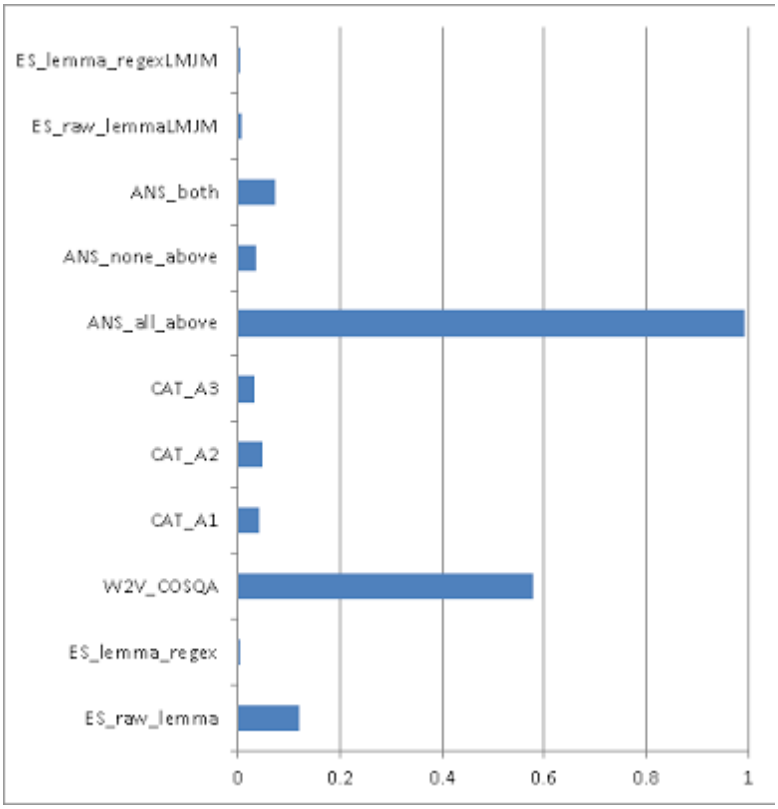
- What were the most important features?

  I've used 22 features in total (11 for each Question/Answer pair and then pairwise combinations between these for each question):

    - **ES_raw_lemma**: IR scores by using ES and raw/lemmatized KB.
    - **ES_lemma_regex**: Regex scoring (number of characters matched) after IR results.
    - **W2V_COSQA**: Cosine similarity between question and answer embeddings.
    - **CAT_A1**: Is multiphrase question + short response?
    - **CAT_A2**: Is fill the _____ + no direct question + long response?
    - **CAT_A3**: Is multiphrase question + long response?
    - **ANS_all_above**: Is *"all of the above"* answer?
    - **ANS_none_above**: Is *"none of the above"* answer?
    - **ANS_both**: Is *"both X and Y"* answer?
    - **ES_raw_lemmaLMJM**: IR scores by using ES and raw/lemmatized KB with LMJM scoring.

- **ES_lemma_regexLMJM**: Regex scoring (number of characters matched) after IR results using LMJM.

The scaled LR coefficients show that both corpus heuristics and embedding-based features have the highest weight, followed by combined IR scores.



- How did you select features?

  The best features were selected by using cross-validation and also manual analysis of the training data.

- Did you make any important feature transformations?

  My data pipeline was the following:

  - Noise reduction (question boilerplate removal)
  - Stopword removal (standard English NLTK stopwords)
  - Lemmatization (NLTK Wordnet Lemmatizer)

```
def procline(q):
    q=q.replace('Which of the following ','').replace('Which of these ','')
    q=q.replace('Were do ','').replace('If a ','').replace('In which ','').
    q=q.replace('What best ','').replace('What is ','').replace('What would
    q=q.replace('Why is ','').replace('Which ','')
    q = q.replace('\t',' ').replace('\n',' ').replace('\r', ' ').replace('
    q = re.sub('[^a-zA-Z0-9\- \']+', " ", q)
    q=[lemmaorstem(m)  for m in q.lower().split(' ') if m not in set_stopwo
    return ' '.join(q)
```

- Did you find any interesting interactions between features?

  I've found a huge performance increase when using a 3-class classification approach, rather than using a 2-class or 4-class one. This third class was used to distribute the probabilities over the remaining two questions.

- Did you use external data?

  The KB was indexed in ES and was automatically generated by querying the Quizlet API for a fixed set of science topic keywords. I've also added the CK-12 workbooks published in the forum. This was a one-off process so no information from the training or testing set was used in order to build the KB. The machine learning model is fully standalone and does not require any additional external data in order to generate predictions.

  I've also used the following multi-state question/answer datasets in order to augment the original training data:

  - Regents 4 grade QA dataset (from Aristo website)
  - Multi-State QA dataset (from Aristo website)

3. Training Method(s)

- What training methods did you use?

  Unsupervised: Gensim's Word2Vec in order to extract embeddings from the KB.

Supervised: 3-class Logistic Regression over IR, W2V and heuristic/statistical corpus features. Local CV scores (5-fold) were around 0.59-0.61 depending on the feature set.

- Did you ensemble the models?

  I've tried ensembling XGBoost and RNN models but nothing could beat the simpler linear model.

4. Interesting findings

- What was the most important trick you used?

  I jumped to the first position after treating the QA challenge as a question-answer pair ranking problem. I've also expanded the Word2Vec model vocabulary by using fuzzy string matching. It helped me to improve the score during the first stage of the competition.

- What do you think set you apart from others in the competition?

  I think my knowledge base was quite good in terms of coverage. I've found Wikipedia IR models to be broader but also noisier.

- Things that I tried and didn't improve my score:

  I couldn't get anything useful out of state of the art RNNs or CNNs. I've also tried to use resources such as ConceptNet or WordNet in order to augment my KB or improve the quality of the embeddings. However, none of these experiments gave me any substantial improvement.

5. Background

- What was your background prior to entering this challenge?

  My Ph.D. topic is related to social media text analysis but for the last 4 years I have been working in cyber-security analytics.
- Did you have any prior experience that helped you succeed in this competition?

Even though I have an NLP background I haven't had much exposure to QA tasks before. I think that being familiar with the right tools is always an advantage, especially when you deal with difficult problems like this one.

- What made you decide to enter this competition?

  I've missed most of the previous NLP-related competitions in Kaggle so I thought that was a good idea to participate in this one.

- How much time did you spend on the competition?

  Around one hour per day (on average).