Documentation: KeyWord Extraction

In this we are trying to find the keyword which can represent the whole text. Till now we have used two algorithms:

1. **Using Rake-nltk package:** In this we have used rake package of the python.
2. **KeyWord Finder:** In this we have used Logistic Regression model. For each observation of training set all keywords associated with the article are termed as positive examples and rest are termed as negative examples.

Features that we have used are:

a) Frequency of words: TF, TF-IDF score, Wiki Freq.
b) Structure of word: Term length, capitalisation
c) Type of word: Named Entity, Noun Phrases, trigram
d) Relationship between word and input text: First occurance in text, Distance between occurance in text.

For each keyword classifier examines 10 associated features and assigns a probability *p* of the candidate being a keyword. If p>0.75 then candidate is selected as a keyword.

3. **Embeded:** This is an unsupervised model in which we are using Doc2vec to assign the vectors to both the document and keyphrase.

a) Phrases that consist of any number of adjectives and one or more nouns are considered to be the candidate keyphrases.
b) Now we use doc2vec to find the cosine similarity between document and candidate keyphrase.
c) Now we rank keyphrases using Maximal Marginal Relevance (MMR) to avoid redundancy problem.

$$\textbf{MMR} := \arg\max\left[\lambda \cdot cos_{sim}(C_i, doc) - (1-\lambda)\max cos_{sim}(C_i, C_j)\right]$$

When λ = 1 MMR computes a standard, relevance ranked list, while when λ = 0 it computes a maximal diversity ranking of the documents in R. So we will keep λ = 0.5, so that relevance and diversity parts of the equation have equal importance.

Please refer paper for further information(EmbedRank: Unsupervised Keyphrase Extraction using Sentence Embeddings)

We have used precision, recall and f1-score as a measure to compare between algorithms. The code that we have used is:

```
tp = 0
fp = 0
fn = 0

for k in y_pred:
      if k in y_true:
          tp += 1
      elif k not in y_true:
          fp += 1

for k in y_true:
      if k not in y_pred:
          fn += 1
p = tp * 1.0 / (tp + fp)
re = tp*1.0/(tp+fn)
f1 = 2*p*re*1.0/(p + re)
```

We finally took the average of all the precisions, recalls and f1-scores from each algorithm

Accurcy and Computation time for the above algorithm taking Crowd500 dataset are:

Method	Precision	Recall	F1-score	Time	Dataset
Embeded	0.2779	0.3421	0.2906	578.2	(Crowd)
Rake	0.1896	0.2087	0.1493	0.41	(Crowd)
Lavanya Sharan KE	0.7555	0.1697	0.2418	52	(Crowd)