Using Statistical and Semantic Models for Multi-Document Summarization

Abstract

We report a series of experiments with different semantic models on top of various statistical models for extractive text summarization. Though statistical models may better capture word co-occurrences and distribution around the text, they fail to detect the context and the sense of sentences /words as a whole. Semantic models help us gain better insight into the context of sentences. We show that how tuning weights between different models can help us achieve significant results on various benchmarks. Learning pre-trained vectors used in semantic models further, on given corpus, can give addition spike in performance. Using weighing techniques in between different statistical models too further refines our result. For Statistical models, we have used TF/IDF, TextRAnk, Jaccard/Cosine Similarities. For Semantic Models, we have used WordNet-based Model and proposed two models based on Glove Vectors and Facebook's InferSent. We tested our approach on DUC 2004 dataset, generating 100-word summaries. We have discussed the system, algorithms, analysis and also proposed and tested possible improvements. ROUGE scores were used to compare to other summarizers.

Introduction

Automatic Text Summarization deals with the task of condensing documents into a summary, whose level is similar to a human-generated summary. It is mostly distributed into two distinct domains, i.e., Abstractive Summarization and Extractive Summarization. Abstractive summarization(Dejong et al. ,1978) involves models to deduce the crux of the document. It then presents a summary consisting of words and phrases that were not there in the actual document, sometimes even paraphrasing BIBREF1 . A state of art method proposed by Wenyuan Zeng BIBREF2 produces such summaries with length restricted to 75. There have been many recent developments that produce optimal results, but it is still in a developing phase. It highly relies on natural language processing techniques, which is still evolving to match human standards. These shortcomings make abstractive summarization highly domain selective. As a result, their application is skewed to the areas where NLP techniques have been superlative. Extractive Summarization, on the other hand, uses different methods to identify the most informative/dominant sentences through the text, and then present the results, ranking them accordingly. In this paper, we have proposed two novel stand-alone summarization methods.The first method is based on Glove Model BIBREF3 ,and other is based on Facebook's InferSent BIBREF4 . We have also discussed how we can effectively subdue shortcomings of one model by using it in coalition with models which capture the view that other faintly held.

Related Work

A vast number of methods have been used for document summarization. Some of the methods include determining the length and positioning of sentences in the text BIBREF5 , deducing centroid terms to find the importance of text BIBREF5 and setting a threshold on average TF-IDF scores. Bag-of-words approach, i.e., making sentence/Word freq matrix, using a signature set of words and assigning them weights to use them as a criterion for importance measure BIBREF6 have also been used. Summarization using weights on high-frequency words BIBREF7 describes that high-frequency terms can be used to deduce the core of document.

While semantic summarizers like Lexical similarity is based on the assumption that important sentences are identified by strong chains BIBREF8 , BIBREF9 , BIBREF10 . In other words, it relates sentences that employ words with the same meaning (synonyms) or other semantic relation. It uses WordNet BIBREF11 to find similarity among words that apply to Word Frequency algorithm.POS(Part of Speech) Tagging and WSD(Word Sense Disambiguation) are common among semantic summarizers. Graphical summarizers like TextRank have also provided great benchmark results.TextRank assigns weights to important keywords from the document using graph-based model and sentences which capture most of those concepts/keywords are ranked higher) BIBREF9 , BIBREF12 TextRank uses Google's PageRank (Brin and Page, 1998) for graphical modeling. Though semantic and graphical models may better capture the sense of document but miss out on statistical view.

There is a void of hybrid summarizers; there haven't been many studies made in the area.Wong BIBREF13 conducted some preliminary research but there isn't much there on benchmark tests to our knowledge. We use a mixture of statistical and semantic models, assign weights among them by training on field-specific corpora. As there is a significant variation in choices among different fields. We support our proposal with expectations that shortcomings posed by one model can be filled with positives from others. We deploy experimental analysis to test our proposition.

Proposed Approach

For Statistical analysis we use Similarity matrices, word co-occurrence/ n-gram model, andTF/IDF matrix. For semantic analysis we use custom Glove based model, WordNet based Model and Facebook InferSent BIBREF4 based Model. For Multi-Document Summarization,after training on corpus, we assign weights among the different techniques .We store the sense vector for documents, along with weights, for future reference. For Single document summarization, firstly we calculate the sense vector for that document and calculate the nearest vector from the stored Vectors, we use the weights of the nearest vector. We will describe the flow for semantic and statistical models separately.

Prepossessing

We discuss, in detail, the steps that are common for both statistical and semantic models.

We use NLTK sentence tokenizer sent_tokenize(), based on PUNKT tokenizer, pre-trained on a corpus. It can differentiate between Mr. , Mrs. and other abbreviations etc. and the normal sentence boundaries. BIBREF14

Given a document INLINEFORM0 we tokenize it into sentences as < INLINEFORM1 >.

Replacing all the special characters with spaces for easier word-tagging and Tokenizing.

We use NLTK word tokenizer, which is a Penn Treebank–style tokenizer, to tokenize words.We calculate the total unique words in the Document. If we can write any sentence as:-

INLINEFORM0 < INLINEFORM1 >, INLINEFORM2

Then the number of unique words can be represented as:- INLINEFORM0 INLINEFORM1

Using Stastical Models

paragraph4 3.25ex plus1ex minus.2ex -1em Frequency Matrix generation: Our tokenized words contain redundancy due to digits and transitional words such as “and”, “but” etc., which carry little information. Such words are termed stop words. BIBREF15 We removed stop words and words occurring in <0.2% and >15% of the documents (considering the word frequency over all documents). After the removal, the no. of unique words left in the particular document be p where p<m (where m is the total no. of unique words in our tokenized list originally). We now formulate a matrix INLINEFORM0 where n is the total number of sentences and p is the total number of unique words left in the document. Element INLINEFORM1 in the matrix INLINEFORM2 denotes frequency of INLINEFORM3 unique word in the INLINEFORM4 sentence.

paragraph4 3.25ex plus1ex minus.2ex -1em Similarity/Correlation Matrix generation: We now have have sentence word frequency vector INLINEFORM0 as < INLINEFORM1 > where INLINEFORM2 denotes frequency of INLINEFORM3 unique word in the INLINEFORM4 sentence. We now compute, INLINEFORM5

We use two similarity measures :

Jaccard Similarity

Cosine Similarity

We generate the similarity matrix INLINEFORM0 for each of the similarity Measure, where INLINEFORM1 indexes the similarity Measure. Element INLINEFORM2 of INLINEFORM3 denotes similarity between INLINEFORM4 and INLINEFORM5 sentence. Consequentially, we will end up with INLINEFORM6 and INLINEFORM7 , corresponding to each similarity measure.

For some sets A and B, <a,b,c,... >and <x,y,z,... >respectively, the Jaccard Similarity is defined as:- INLINEFORM0

The Cosine distance between `u' and `v', is defined as:- INLINEFORM0

where INLINEFORM0 is the dot product of INLINEFORM1 and INLINEFORM2 .

PageRank algorithm BIBREF16 , devised to rank web pages, forms the core of Google Search. It roughly works by ranking pages according to the number and quality of outsourcing links from the page. For NLP, a PageRank based technique ,TextRank has been a major breakthrough in the field. TextRank based summarization has seeded exemplary results on benchmarks. We use a naive TextRank analogous for our task.

Given INLINEFORM0 sentences < INLINEFORM1 >, we intend to generate PageRank or probability distribution matrix INLINEFORM2 , INLINEFORM3

, where INLINEFORM0 in original paper denoted probability with which a randomly browsing user lands on a particular page. For the summarization task, they denote how strongly a sentence is connected with rest of document, or how well sentence captures multiple views/concepts. The steps are as:

Initialize INLINEFORM0 as, INLINEFORM1

Define INLINEFORM0 , probability that randomly chosen sentence is in summary and INLINEFORM1 as measure of change i.e. to stop computation when difference between to successive INLINEFORM2 computations recedes below INLINEFORM3 .

Using cosine-similarity matrix INLINEFORM0 , we generate the following equation as a measure for relation between sentences:- INLINEFORM1

Repeat last step until INLINEFORM0 .

Take top ranking sentences in INLINEFORM0 for summary.

Term Frequency(TF)/Bag of words is the count of how many times a word occurs in the given document. Inverse Document Frequency(IDF) is the number of times word occurs in complete corpus. Infrequent words through corpus will have higher weights, while weights for more frequent words will be depricated.

Underlying steps for TF/IDF summarization are:

Create a count vector INLINEFORM0

Build a tf-idf matrix INLINEFORM0 with element INLINEFORM1 as, INLINEFORM2

Here, INLINEFORM0 denotes term frequency of ith word in jth sentence, and INLINEFORM1 represents the IDF frequency.

Score each sentence, taking into consideration only nouns, we use NLTK POS-tagger for identifying nouns. INLINEFORM0

Applying positional weighing . INLINEFORM0 INLINEFORM1

Summarize using top ranking sentences.

Using Semantic Models

We proceed in the same way as we did for statistical models. All the pre-processing steps remain nearly same. We can make a little change by using lemmatizer instead of stemmer. Stemming involves removing the derivational affixes/end of words by heuristic analysis in hope to achieve base form. Lemmatization, on the other hand, involves firstly POS tagging BIBREF17 , and after morphological and vocabulary analysis, reducing the word to its base form. Stemmer output for `goes' is `goe', while lemmatized output with the verb passed as POS tag is `go'. Though lemmatization may have little more time overhead as compared to stemming, it necessarily provides better base word reductions. Since WordNet BIBREF18 and Glove both require dictionary look-ups, in order for them to work well, we need better base word mappings. Hence lemmatization is preferred.

Part of Speech(POS) Tagging: We tag the words using NLTK POS-Tagger.

Lemmatization: We use NTLK lemmatizer with POS tags passed as contexts.

We generated Similarity matrices in the case of Statistical Models. We will do the same here, but for sentence similarity measure we use the method devised by Dao. BIBREF19 The method is defined as:

Word Sense Disambiguation(WSD): We use the adapted version of Lesk algorithm BIBREF20 , as devised by Dao, to derive the sense for each word.

Sentence pair Similarity: For each pair of sentences, we create semantic similarity matrix INLINEFORM0 . Let INLINEFORM1 and INLINEFORM2 be two sentences of lengths INLINEFORM3 and INLINEFORM4 respectively. Then the resultant matrix INLINEFORM5 will be of size INLINEFORM6 , with element INLINEFORM7 denoting semantic similarity between sense/synset of word at position INLINEFORM8 in sentence INLINEFORM9 and sense/synset of word at position INLINEFORM10 in sentence INLINEFORM11 , which is calculated by path length similarity using is-a (hypernym/hyponym) hierarchies. It uses the idea that shorter the path length, higher the similarity. To calculate the path length, we proceed in following manner:-

For two words INLINEFORM0 and INLINEFORM1 , with synsets INLINEFORM2 and INLINEFORM3 respectively, INLINEFORM4 INLINEFORM5

We formulate the problem of capturing semantic similarity between sentences as the problem of computing a maximum total matching weight of a bipartite graph, where X and Y are two sets of disjoint nodes. We use the Hungarian method BIBREF21 to solve this problem. Finally we get bipartite matching matrix INLINEFORM0 with entry INLINEFORM1 denoting matching between INLINEFORM2 and INLINEFORM3 . To obtain the overall similarity, we use Dice coefficient, INLINEFORM4

with threshold set to INLINEFORM0 , and INLINEFORM1 , INLINEFORM2 denoting lengths of sentence INLINEFORM3 and INLINEFORM4 respectively.

We perform the previous step over all pairs to generate the similarity matrix INLINEFORM0 .

Glove Model provides us with a convenient method to represent words as vectors, using vectors representation for words, we generate vector representation for sentences. We work in the following order,

Represent each tokenized word INLINEFORM0 in its vector form < INLINEFORM1 >.

Represent each sentence into vector using following equation, INLINEFORM0

where INLINEFORM0 being frequency of INLINEFORM1 in INLINEFORM2 .

Calculate similarity between sentences using cosine distance between two sentence vectors.

Populate similarity matrix INLINEFORM0 using previous step.

Infersent is a state of the art supervised sentence encoding technique BIBREF4 . It outperformed another state-of-the-art sentence encoder SkipThought on several benchmarks, like the STS benchmark (http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark). The model is trained on Stanford Natural

Language Inference (SNLI) dataset BIBREF22 using seven architectures Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), forward and backward GRU with hidden states concatenated, Bi-directional LSTMs (BiLSTM) with min/max pooling, self-attentive network and (HCN's) Hierarchical convolutional networks. The network performances are task/corpus specific.

Steps to generate similarity matrix INLINEFORM0 are:

Encode each sentence to generate its vector representation < INLINEFORM0 >.

Calculate similarity between sentence pair using cosine distance.

Populate similarity matrix INLINEFORM0 using previous step.

Generating Summaries

TF-IDF scores and TextRank allows us to directly rank sentences and choose INLINEFORM0 top sentences, where INLINEFORM1 is how many sentences user want in the summary. On the other hand, the similarity matrix based approach is used in case of all Semantic Models, and Similarity/correlation based Statistical models. To rank sentences from Similarity matrix, we can use following approaches:-

Ranking through Relevance score

For each sentence INLINEFORM0 in similarity matrix the Relevance Score is as:-

INLINEFORM0

We can now choose INLINEFORM0 top ranking sentences by RScores. Higher the RScore, higher the rank of sentence.

Hierarchical Clustering

Given a similarity matrix INLINEFORM0 , let INLINEFORM1 denote an individual element, then Hierarchical clustering is performed as follows:-

Initialize a empty list INLINEFORM0 .

Choose element with highest similarity value let it be INLINEFORM0 where, INLINEFORM1

Replace values in column and row INLINEFORM0 in following manner:-

INLINEFORM0

INLINEFORM0

Replace entries corresponding to column and row INLINEFORM0 by zeros.

Add INLINEFORM0 and INLINEFORM1 to INLINEFORM2 , if they are not already there.

Repeat steps 2-5 until single single non-zero element remains, for remaining non-zero element apply Step 5 and terminate.

We will have rank list INLINEFORM0 in the end.

We can now choose INLINEFORM0 top ranking sentences from INLINEFORM1 .

Single Document Summarization

After generating summary from a particular model, our aim is to compute summaries through overlap of different models. Let us have INLINEFORM0 summaries from INLINEFORM1 different models. For INLINEFORM2 summarization model, let the INLINEFORM3 sentences contained be:-

INLINEFORM0

Now for our list of sentences INLINEFORM0 we define cWeight as weight obtained for each sentence using INLINEFORM1 models.

INLINEFORM0

Here, INLINEFORM0 is a function which returns 1 if sentence is in summary of INLINEFORM1 model, otherwise zero. INLINEFORM2 is weight assigned to each model without training, INLINEFORM3

Multi-Document/Domain-Specific Summarization

We here use machine learning based approach to further increase the quality of our summarization technique. The elemental concept is that we use training set of INLINEFORM0 domain specific documents, with gold standard/human-composed summaries, provided we fine tune our weights INLINEFORM1 for different models taking F1-score/F-measure. BIBREF23 as factor. INLINEFORM2

We proceed in the following manner:-

For each document in training set generate summary using each model independently, compute the INLINEFORM0 w.r.t. gold summary.

For each model, assign the weights using INLINEFORM0

Here, INLINEFORM0 denotes INLINEFORM1 for INLINEFORM2 model in INLINEFORM3 document.

We now obtain cWeight as we did previously, and formulate cumulative summary, capturing the consensus of different models. We hence used a supervised learning algorithm to capture the mean performances of different models over the training data to fine-tune our summary.

Domain-Specific Single Document Summarization

As we discussed earlier, summarization models are field selective. Some models tend to perform remarkably better than others in certain fields. So, instead of assigning uniform weights to all models we can go by the following approach.

For each set of documents we train on, we generate document vector using bidirectional GRU (BIBREF24 as described by Zichao Yang BIBREF25 for each document. We then generate complete corpus vector as follows:- INLINEFORM0

where, INLINEFORM0 is total training set size, INLINEFORM1 is number of features in document vector.

We save INLINEFORM0 and INLINEFORM1 corresponding to each corpus.

For each single document summarization task, we generate given texts document vector, perform nearest vector search over all stored INLINEFORM0 , apply weights corresponding to that corpus.

Experiments

We evaluate our approaches on 2004 DUC(Document Understanding Conferences) dataset(https://duc.nist.gov/). The Dataset has 5 Tasks in total. We work on Task 2. It (Task 2) contains 50 news documents cluster for multi-document summarization. Only 665-character summaries are provided for each cluster. For evaluation, we use ROGUE, an automatic summary evaluation metric. It was firstly used for DUC 2004 data-set. Now, it has become a benchmark for evaluation of automated summaries. ROUGE is a correlation metric for fixed-length summaries populated using n-gram co-occurrence. For comparison between model summary and to-be evaluated summary, separate scores for 1, 2, 3, and 4-gram matching are kept. We use ROUGE-2, a bi-gram based matching technique for our task.

In the Table 1, we try different model pairs with weights trained on corpus for Task 2. We have displayed mean ROUGE-2 scores for base Models. We have calculated final scores taking into consideration all normalizations, stemming, lemmatizing and clustering techniques, and the ones providing best results were used. We generally expected WordNet, Glove based semantic models to perform better given they better capture crux of the sentence and compute similarity using the same, but instead, they performed average. This is attributed to the fact they assigned high similarity scores to not so semantically related sentences. We also observe that combinations with TF/IDF and Similarity Matrices(Jaccard/Cosine) offer nearly same results. The InferSent based Summarizer performed exceptionally well. We initially used pre-trained features to generate sentence vectors through InferSent.

Conclusion/Future Work

We can see that using a mixture of Semantic and Statistical models offers an improvement over stand-alone models. Given better training data, results can be further improved. Using domain-specific labeled data can provide a further increase in performances of Glove and WordNet Models.

Some easy additions that can be worked on are:

Unnecessary parts of the sentence can be trimmed to improve summary further.

Using better algorithm to capture sentence vector through Glove Model can improve results.

Query specific summarizer can be implemented with little additions.

For generating summary through model overlaps, we can also try Graph-based methods or different Clustering techniques.