Automatic Text Summarizing using graph based algorithm

Mohamed Farhan
mfarhan@knights.ucf.edu

Mithun Mohanraj
mohanmithun005@knights.ucf.edu

Pushkar
pushkar1992m@knights.ucf.edu

1 INTRODUCTION

1.1 Motivation

Summarization technologies are popular and are in demand due to large volumes of textual information available on the internet. Automatic text summarization techniques have been deployed in multiple natural language applications such as generating short and concise summaries for long news articles or generating summary plots for movies using reviews. Automatic text summarization[1] involves condensing a document or a set of documents to produce a coherent, logical and comprehensible summary. Summarization technologies are very popular and are in demand due to large volumes of textual information available on the internet. In our project, we aim at implementing an unsupervised method for automatic sentence extraction using a graph-based ranking algorithm like Text-Rank. Our aim is to generate a set of logically coherent sentences from a huge document that abridges the content of the document.

1.2 Problem Statement

The objective of our project is to compare the performance of system generated text with reference text using different state-of-the-art embeddings to represent sentences.In the later sections, we draw conclusions over which embeddings performed the best and interpret the results in both quantitative and qualitative manner.

We achieve this objective by working our way through the movie critic reviews provided by Rotten Tomatoes. In our project, the system generated text is obtained through extractive summarization method. In this method, sentences are ranked using a graph based model or architecture whose goal is to rank sentences in text using averaged similarity measure scores between different pairs of sentences.

1.3 Summary of work completed

In this paper we discussed the methodology adopted to address the problem statement mentioned in the previous section, metrics used to evaluate and validate our approach and draw conclusive results both quantitatively and qualitatively.

We start off by generating embeddings (discussed in later sections) for sentences, compute similarity scores and construct the resulting similarity matrix which is used as input for the Text-Rank block. The Text-Rank block produces the system generated text which is compared with the reference text in the evaluation module to determine scores.

Section 2 describes related work that we used to build our project. Section 3 covers detailed methodology adopted to implement our approach.

2 RELATED WORK

2.1 Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization

Graph-based ranking algorithm [3] is a way of deciding on the importance of a vertex within a graph, by considering global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. In this paper[3], the author investigated several graph-based ranking algorithms and evaluated their application to unsupervised sentence extraction in a context. [3]This paper concludes that Text-Rank performed well as its architecture does not depend on extracting information from the local context, rather efforts are made to capture information recursively using complete understanding of the graph. Text-Rank recognizes links between different entities in a document using graphs constructed and thereby recommends sentences capturing the essence of the document. Sentences that imply similar meanings are recommended as being useful for understanding the source text. Such sentences are also assumed to be more informative and are likely to be assigned a higher score while ranking them. Text-Rank's architecture does not require any domain specific information as it works through unsupervised learning. This makes it easier to deploy Text-Rank to other domains without having to worry about the texts deep linguistic structure.

2.2 The Evaluation of Sentence Similarity Measures

In this paper [6], several text similarity measures have been used to calculate similarity score between sentences in many text applications. The author of this paper emphasizes that the correct similarity judgment should be made even if the sentences do not share similar surface form. The paper delves deep into understanding the performance of various similarity measures evaluated on several classes using multiple data sets containing sentence pairs. They also work well with data sets that have high complexity but at the same time do not improve performance on testing data that depends on its characteristics. Measures that involve overlap tend to generate reasonable result. Measures that depend on linguistics do not perform well while evaluating pairs that are highly dissimilar in data sets involving high complexity which drastically affect overall accuracy. Word overlap measures cannot distinguish between dissimilar pairs of sentences because ratio of overlap is very small in data involving high complexity. The author recommends graph based approach for representation instead of sticking to bag of words representation while representing a sentence.

2.3 Text-Rank - Bringing Order into Texts

The author discusses Text-Rank[4] : a graph-based ranking model for text processing, and show how this model can be successfully

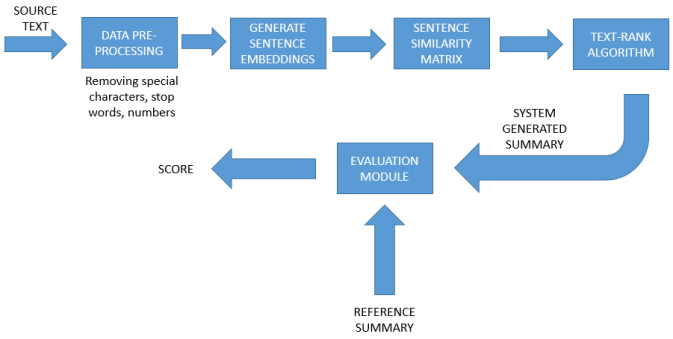

Figure 1: Pipeline demonstrating methodology adopted

used in natural language applications. The paper proposes two innovative unsupervised methods for keyword and sentence extraction, and shows that the results obtained compare favorably with previously published results on established benchmarks. This author of this paper discussed ranking model for processing text which is based on graph architecture and successfully demonstrated that such graph based models can be deployed for natural language processing applications whose objective largely would be to summarize a set of documents or to extract key phrases from source documents. The author elaborated on the two unsupervised techniques adopted for key phrase extraction and text summarization and corroborated with results that Text-Rank produced results that are competitive with results of earlier adopted approaches. The paper emphasis on the fact that Text-Rank does not require any explicit domain information about the textual data making it easier to deploy to other domains.

3 METHODOLOGY

We intend to use Text Rank Algorithm or modified the Page-Rank algorithm [3] to make it compatible with our problem definition which is to rank each sentence in our summary to get the most expressive one. This method can be extrapolated to other domains easily even though we are confining ourselves to using data from movie reviews domain. We used multiple word embedding representations and computed the cosine similarity for each of those representation whose scores were then used to rank sentences of text using modified Page-Rank algorithm. To implement our algorithm we first identified sentences through tokenizing and retrieve all the sentences present in our text document. Also, we ensured that special characters, numerical values and stop words were removed from the source reviews. After that, similarity was measured between different pairs of sentences using cosine similarity[6]. The output of the Text-Rank block gives us the top ranked sentence from each review. Figure 1 describes the pipeline adopted.

3.1 Embeddings

The word embedding representations we used in this project are described in detail in the subsequent subsections.[7]

3.1.1 GloVe Embeddings: GloVe is an unsupervised learning algorithm which learns word representation using corpus level co-occurrence statistics. The resulting representations capture interesting relationships of the word embeddings.[14] GloVe's representation works such that it is capable of capturing global information into account while learning dimensions of meaning. GloVe uses unsupervised learning architecture for word vector representations obtained by learning word to word co-occurrence statistics at the corpus level. Unlike existing unsupervised methods, GloVe tries to create word vectors that can capture semantics among words. GloVe uses ratios of co-occurrence probabilities rather than direct word counts to learn word vectors. The idea uses co-occurrence probabilities to capture certain aspects of meaning in word representations. Relevancy of two words can be examined by their ratio of co-occurrence with various context words. When the ratio is too large, the two words are strongly related to the context word. When the ratio is one, either of two words are not related to the context word.

3.1.2 Word2Vec: Word2Vec is a form of representation that can capture complex meanings. Word2Vec is a powerful tool which generates word embeddings or vectors and can represent complex relationships mathematically.Word2Vec uses a shallow neural network with two layers that can efficiently process text. Given a large corpus as its input, it outputs an equivalent set of vectors such that they represent feature vectors of that corpus for all the words contained. The intuition behind understanding Word2Vec is that it clusters similar words in vector space. It is able to draw parallels between different pair of words mathematically.[9] Also, it is able to build numerical representations taking into consideration the context of words individually.[9] Word2Vec works its architecture such that it can accurately predict the meaning of words based off previous occurrences. It is capable of measuring similarities among words when fed to a deep learning network from a vocabulary.[9] Cosine similarity is adopted to determine how closely the meanings of any two words correspond. Word2Vec is popularly implemented using two methods, where the task in the first method (continuous bag of words) is to predict the target words based on its context words and the task in the second method is to predict a set of context words given a word (skip-gram). We make use of gensim library's pre-trained Word2Vec model on Google's news data set for generating word embeddings for our text.[8]

3.1.3 Character Embeddings: Characters that compose the words can be used to construct representations for the words. Instead of creating character embeddings and combining them in a particular way for word representations, we can construct character enhanced word embeddings obtained from hierarchical embedding architecture[13]. In this architecture, characters are fed sequentially to RNN-based LSTM network . Character-character mapping information is fed to the LSTM layers and output of the LSTM layers is combined with word-word mapping information to produce character enhanced word embeddings. Sentence embeddings are obtained by averaging character enhanced word embeddings for words that compose the sentences. The idea of using character enhanced word embeddings is presented and documented in Flair python package which we make use for generating the embeddings for each sentence in the source text. Character level models perform better than

Word2Vec model because n-grams composed using characters are encountered across all words.[11] Therefore such models can also handle situations where a given word is not found in the vocabulary of a corpus. Such out of vocabulary words can generate their own embeddings on a character level. This is seen as an advantage over word based models like Word2Vec.[11] Also, character level models perform better for words whose frequency is less within the corpus as the n-gram composed characters are encountered across all words. At the atomic level, language models using characters are easily processed by LSTM models where text can be thought to be as sequence of characters.[11] These LSTM models can predict subsequent characters with good accuracy levels.

3.1.4 Flair Embeddings: It generates embeddings using sentences that are passed as sequences of characters into a language model. The resulting embeddings are powerful enough that they capture latent syntactic semantic information that in encoded into them which goes beyond traditional word embeddings. They are trained independently and therefore are able to model words as character sequences. They do a very good job of representing the context efficiently, which is that a same word could have multiple embeddings based on its contextual use.[7][8]

Words are broken down and thought as characters. Its architecture is similar to character level embeddings. Here, the embeddings capture the context of the surrounding or neighboring text more efficiently more so than any other state of the art embedding representations. This implies that a word having multiple meanings is capable of generating different embeddings based on the surrounding context of the text. In the paper [12], Flair embeddings are referred to as contextual string embeddings because these embeddings for any string is obtained based on the context of the sentence in which it is present. Representation for the words is obtained as the output of pre-trained bi-directional character language model. Input to language model is sequence of the characters in a given sentence. The string embeddings are created from internal character states of the language model. [12] This language model is trained without any notion of words and models words as composition of characters. Similar to ELMo embeddings, Flair embeddings for the same word is different based on the context it appears in the corpus. Sentence embeddings are obtained by averaging Flair word embedding for words that compose the sentences. The idea of using flair embeddings is presented and documented in Flair python package which we make use of for generating embeddings for each sentence in the source text.[8]

3.1.5 Bert Embeddings: BERT embeddings developed by Devlin et al. (2018)[16] are a different kind of powerful word embedding representations based on a bidirectional transformer architecture. BERT uses transformers[15], self-attention mechanisms which try to learn the relationship between words in a given sequence. Transformer includes encoders that encodes the input text and decoders that are used for producing the output for a downstream task. The self-attention mechanism is included in both the encoder and decoder. The encoder is bi-directional, meaning that it reads the entire sequence at once and attention is given to words on either side of the center word. The word representations from the encoder output is obtained by learning the context of the word based on its surroundings. In the original paper, it is stated that using the last four hidden layers of encoder for generating contextualized embeddings produces the best results for Named entity recognition. Sentence embeddings are obtained by averaging BERT's contextualized word embedding for words that compose the sentences. BERT embeddings capture the semantics of words based on the bi-directional contextual information obtained through the attention mechanism.

3.1.6 Elmo Embeddings: ELMo embeddings developed by Peters et al. in 2018[17] are using a bidirectional recurrent neural network to predict words in a text.[7][8] ELMo embeddings are deep contextualized word representations obtained from pre-trained bi-directional LSTM for language modelling tasks. ELMo embeddings are constructed from all the internal layers of bi-directional language model which is why they are called deep contextualized word vector representations. In the paper, it is stated that higher level LSTM layers capture word semantics and lower level LSTM layers capture syntactical information. Combining all the internal layers enriches the quality of word representations. They are combined together into a single vector by weighted summation. Sentence embeddings are obtained by averaging ELMoâĂŹs word embeddings for words that compose the sentences. ELMo embeddings generate different embeddings for the same word for different contexts. This means that it allows different embeddings for different sentences even if they share common words. When we calculate similarity measures for sentences represented as ELMo embeddings, it gives us high scores for sentences which convey same semantic information.

3.2 Similarity measure

In our project we use cosine similarity measure to capture the closeness or similarity of each sentence with respect to every other sentence in the text.The cosine similarities between sentences in the text are calculated for different word embedding representation presented earlier in this section. Based on these measures, we construct a similarity matrix for all sentences in the text which is used as an input to Text-Rank algorithm for creating a graph which represents the sentences in the node and similarity measure as edges between the nodes.

$$similarity = cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

3.3 Text Rank algorithm

The central theme of this project is to make use of a modified version of the popular Page-Rank algorithm. Page-Rank was initially used to rank web pages and is popularly used by search engine giant Google. We make use of a slightly modified version of this algorithm to make it suitable for text summarization.

We were able to verify the intuition behind Page Rank algorithm which is being modified to our needs to develop this project. The modified algorithm known as Text Rank algorithm is an unsupervised technique used to generate summarization of texts. In this, the most important words/sentences of a document are extracted using

various similarity measures one such measure being the cosine similarity as discussed earlier.

The intuition behind Text-Rank algorithm is to determine a set of scores for each sentence in a corpus/review and compute the top ranked sentences sorting them based on their scores, which leads us to a more condensed summary of a corpus. For text rank algorithm, sentences are considered equivalent to pages and similarity of two sentences is probability of going from one sentence to other.

The performance of the algorithm strongly depends on similarity measure being used.The value for similarity measure is between [0,1]. A good similarity measure will give 1 if the objects are equal and 0 if objects are mutually exclusive.

Text-Rank is a graph-based algorithm which calculates the importance of the text based on global information obtained in a recursive way from the graph itself. The idea behind finding the importance of the sentence is that it maps each sentence in the text to a vertex and similarity scores between the sentences are taken as edge weights.[10] The ranking of vertex is based on voting. When one vertex links to other, it castes a vote to the other vertex. Rank of the vertex depends on the number of votes it gets and importance of the vertex casting the vote. In case of sentence ranking, highest ranked sentences are the ones which have higher similarity scores on average to all the other sentences in the graph. This in fact forms the basis for using graph-based algorithm for extractive summarization to use top m ranked sentences as the summary of the source text.[10]

Something interesting to note here is that the Text-Rank algorithm will generate significantly different summaries for different similarity scores. Therefore, care should be taken to ensure that the similarity measures used to construct the scores matrix fits well with the model.

4 DATASET AND EVALUATION METRICS

4.1 Dataset

We plan on using a data-set which is a set of reviews collected from Rotten Tomatoes[5]. It contains relevant information such as movie id, movie name, Rotten Tomato one-line summary, critic reviews of movies. Rotten Tomatoes is a good resource for our project as it is an online movie review database which contains reviews by movie critics and audiences.

4.2 Data collection

Rotten Tomatoes is a good resource to collect information on movies as it is a well-established database that houses such information. It contains relevant information on movies including reviews from audiences and critics. Therefore, we decided to create our data set using Rotten Tomatoes.

We collected our data from Rotten Tomatoes using a web scrapper. This scrapper visits the Rotten Tomatoes homepage and finds its way to the critics review section for all the different. It then extracts the one line summary of the review which we use later for evaluation of the model generated movie summary. It also gets the comprehensive review of the movie. All of this data is then written to a file along with the movie name and movie id.

4.3 Data pre-processing

After collecting data, we had to clean the data as most of it required some form of pre-processing. We had to tokenize the reviews into sentences. We also set a threshold to make sure that the cleaned data set only contained reviews with at least 50 words. Therefore we had to discard reviews with less than 50 words.

We completed this task using the powerful tools available to us in the nltk library. These sentences were tokenized further into individual words so that it would be easier to capture better representation of words. All the different numerical values that were encountered in the reviews were discarded along with punctuation and non-unicode characters.

4.4 ROUGE Evaluation

ROUGE[2] is a set of metrics used for evaluating automatic summarization in natural language processing. The metrics compare an automatically produced summary against a reference or a set of references (human-produced) summary or translation.

ROUGE-N- measures uni-gram, bi-gram, trigram and higher order n-gram overlap between the reference and system generated summary.This metric measures how much information of the reference summary is captured in system generated summary. ROUGE-L-measures longest matching sequence of words between reference and system generated summary using LCS (longest common subsequence).LCS captures in-sequence word matches which convey certain aspects of sequence level word order.

$$ROGUE\ recall = \frac{number\ of\ overlapping\ words}{total\ words\ in\ reference\ summary} \quad (2)$$

$$ROGUE\ precision = \frac{number\ of\ overlapping\ words}{total\ words\ in\ system\ summary} \quad (3)$$

Precision is calculated as the fraction of number of n-grams in Rotten Tomatoes one-line summary. It is assumed that the one line summaries picked up from Rotten Tomatoes are ideal human generated summaries.They are used as a reference during the computation of ROUGE scores. For ROUGE-N , number of overlapping n-grams between RottenTomatoes summaries and summaries generated using the model is calculated.

Precision is calculated as the fraction of number of over-lapping n-grams in system generated summary.Precision captures how much system generated summaries are concise in nature. Recall is fraction of number of over-lapping n-grams in the reference summary. Recall measures how much of the reference summary is captured in the system generated summary. We calculate the F-1 score using precision and recall. These evaluations are calculated for all different word embedding representations . Mean ROUGE-N and ROUGE-L scores is used as a measure to compare the performance of different embeddings.

5 EVALUATION RESULTS

Table 1 shows the averaged ROUGE-1 , ROUGE-2 and ROUGE-L and F1 Scores for all the embeddings on 30 movie reviews. In context of ROUGE, F1 score measures the ability of generated summaries, capturing the information from the source text in a concise manner.

Table 1: F1 score

Embeddings	ROUGE-1	ROUGE-2	ROUGE-L
BERT	0.1020	0.015	0.067
Character	0.1089	0.019	0.0745
ELMO	0.1050	0.02	0.073
Flair	0.1130	0.02	0.077
GloVe	0.0110	0.01	0.008
Word2Vec	0.008	0.01	0.007

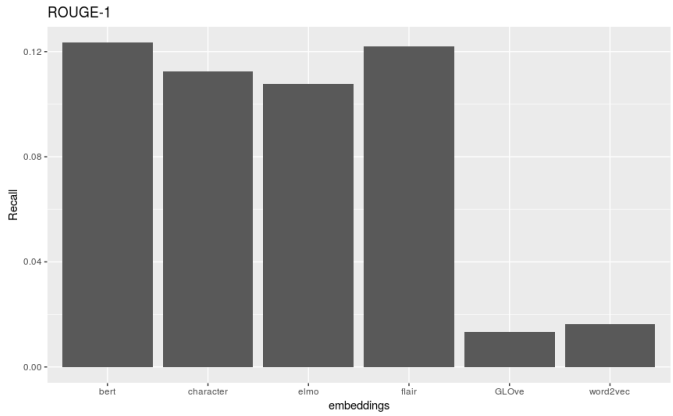

Figure 2: Plot showing averaged ROUGE-1 Recall scores for different embeddings

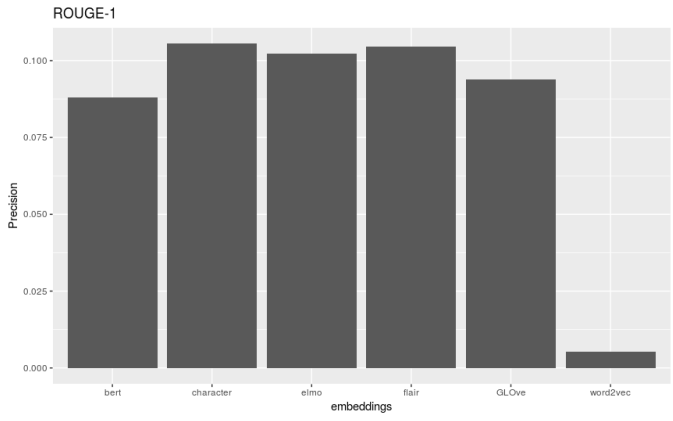

Figure 3: Plot showing averaged ROUGE-1 Precision scores for different embeddings

From table 1, we can infer that BERT, Character, ELMo and Flair have reasonably high f-1 scores for all the metrics compared to the other embeddings.

6 DISCUSSION

Rouge-2 scores measures the number of overlapping bi-grams between the reference and system generated summaries.Based on the results obtained, we can argue that Flair, Character and ELMo embeddings generate fluent summaries as they have high ROUGE-2 scores compared to other embeddings.Intuitively, high over-lap of bi-grams suggests that the summaries capture syntactic information. We find that F-1 scores are low overall as reference summary is abstractive in nature and contains only few words from source review. In figure 2, we see that BERT embeddings have best average recall scores, indicating that they capture more information from source review. We also observed that Character level embeddings generated better average precision scores from the figure 3 indicating that they produce short and crisp summaries.

7 RISK MANAGEMENT

Evaluating summarization models is challenging as determining what constitutes a perfect summary of a review varies among different people.Sentences generated by our model may contain mixture of relevant, non-relevant information which sometimes may contain partially redundant information.

Also, comparing our system generated summaries against Rotten Tomatoes one-line summaries may not be the best form of evaluation as these one-line summaries might represent the best sentence that captures the essence of the review.

Another baseline that could be adopted is to compare the system generated summaries against the first and/or last sentences of the entire review as it is highly likely that these sentences might best represent the essence of the review.

The above mentioned limitation can be overcome using abstractive summarization (supervised) technique which we plan on implementing if time permits.

8 CONCLUSION

Flair embeddings performed well overall based on F-1 scores.But the f-1 scores for Flair, BERT,ELMo and Character embeddings are comparatively same for all the Rouge metrics.This implies that all these embeddings performs reasonably well for capturing more information from the source text compared to GloVe and Word2Vec. The reference summaries in data-set were abstractive in nature making it difficult to effectively evaluate system generated summaries and different embeddings For future work, we consider using different similarity measures and evaluation metrics that can effectively compare the performance of generated summaries.

9 WORK DISTRIBUTION

1.)Pushkar- Code for evaluation, Report write-up (Related Works, Evaluation, Discussion, Conclusion)

2.)Mithun Mohanraj- Code for generating various embeddings, Text- Rank algorithm, Generating summaries.

3.)Mohamed Farhan- Preprocessing data, Slides preparation, Report write-up (Introduction, Problem Statement, Methodology)

Link to the code:- https://www.dropbox.com/sh/nlgcqadioco1i5n/AABWQCO8o76aF$_Z SKrJESC7ha?dl = 0$

10 REFERENCES

1.)A Survey on Automatic Text Summarization-https://www.cs.cmu.edu/ afm/Home-files/Das-Martins-survey-summarization.pdf

2.)Extractive Summarization Using Supervised and Semi-Supervised Learning-http://anthology.aclweb.org/C/C08/C08-1124.pdf

3.)Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization - http://www.aclweb.org/anthology/P04-3020

4.)TextRank: Bringing Order into Texts - https://web.eecs.umich.edu/ mihalcea/papers/mihalcea.emnlp04.pdf

5.)Movie Review Mining and Summarization -https://pdfs.semanticscholar.org/d576/d9ea5cc898d2fb4e833a630e59ff02edc7a8.pdf

6.)The Evaluation of Sentence Similarity Measures http://www.cis.drexel.edu/faculty/thu/research-papers/dawak-547.pdf

7.)Contextual String Embeddings for Sequence Labeling. Alan Akbik, Duncan Blythe and Roland Vollgraf. 27th International Conference on Computational Linguistics, COLING 2018

8.)Flair package used in code - https://github.com/zalandoresearch/flair

9.)Word2Vec - https://skymind.ai/wiki/word2vec

10.)Text-Rank algorithm - https://web.eecs.umich.edu/ mihalcea/papers/mihalcea.emnlp04.pdf

11.) Character level embeddings - http://www.davidsbatista.net/blog/2018/12/06/

12.) Flair embeddings - https://drive.google.com/file/d/17yVpFA7MmXaQFTe-HDpZuqw9fJlmzg56/view?usp=sharing

13.)http://ne uroner.com/NeuroNERengine$_{w}ith_{c}aption_{n}o_{f}igure.png$

14.)Glove embeddings-https://nlp.stanford.edu/pubs/glove.pdf

15.)https://www.kdnuggets.com/2018/12/bert-sota-nlp-model-explained.html

16.)BERT embeddings-https://arxiv.org/abs/1810.04805

17.)ELMO embedings-https://www.aclweb.org/anthology/N18-1202

18.)Rouge-http://www.ccs.neu.edu/home/vip/teach/DMcourse/5$_{t}opicmodel_{s}um$
$is - ROUGE.pdf$