Word Embeddings To Document Distances And Summaries

Lab Report: Development and Application of Data Mining and Learning Systems 2016

Vishwani Gupta
Supervisor: Sven Giesselbach

The goal of this report is to discuss and explore different heuristics to generate summaries of a set of documents using Word2Vec and Word Mover Distance. Word2Vec proposed by Tomas Mikolov et al. and Word Mover Distance Kusner et al. have also been discussed in detail. Word Mover Distances uses word embeddings generated from Word2Vec. Comparisons and challenges faced during implementation of Word Mover Distance algorithm and optimization techniques are also been discussed in detail. The main point of using the Word Mover Distance in generating summaries is that it captures semantic similarities between words. In this way, we can extract sentences related to general idea and also eliminate those which are not.

1. Introduction

The goal of this lab is to generate summaries which uses Word Mover Distance. To generate Word Mover Distance, the first task is to build a Word2Vec model using different text corpora and compare them to determine which gives the best results. This model helps in generating word vectors needed for the next step which is to implement Word Mover Distance. It has been seen that since Google pre-trained model makes use of a huge training data, it is best suited to generate Word Mover Distance. Therefore Word Mover Distance algorithm is implemented in Python, using Google pre-trained model.
There are three different optimizing techniques proposed by Kusner et al. [2015]. The three optimization techniques are explored and compared for the time they take for computing distances between a set of documents.

Using the result of Word2Vec and Word Mover Distance algorithm, different heuristics for generating a summary of a set of documents are explored. The first step is to generate a general idea about the text using Word Mover Distance. Next step is to make sure that any other detail in the documents is not missed. Together with new details, general idea will form the summary of the documents.

2. Related Work

2.1. Word2Vec Algorithm

The model is proposed by Mikolov et al. [2013b]. It is an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data and outputs a vector representation of words based on the context of the words in the training data. Two approaches have been proposed for generating word vectors: Continuous Bag of Words(CBOW) and Skip Gram Model. These both are quite efficient and easy to train when compared to the dense neural network since they involve only one hidden layer.

One of the most intriguing fact about the model is that learned vectors explicitly encode many linguistic regularities and patterns. For example, the result of a vector calculation Vec(Berlin) - Vec(Germany) + Vec(France) is closer to Vec(Paris) than to any other word vector, which is correct since they are Capitals of the respective nations.

2.1.1. Continuous Bag of Words(CBOW)

Continuous Bag of Words(CBOW) is proposed by Mikolov et al. [2013a]. The goal of the model to predict one target word based on the input context words. The input is , therefore, a set of words which makes the context and output is a word related to the context. As explained by Xin

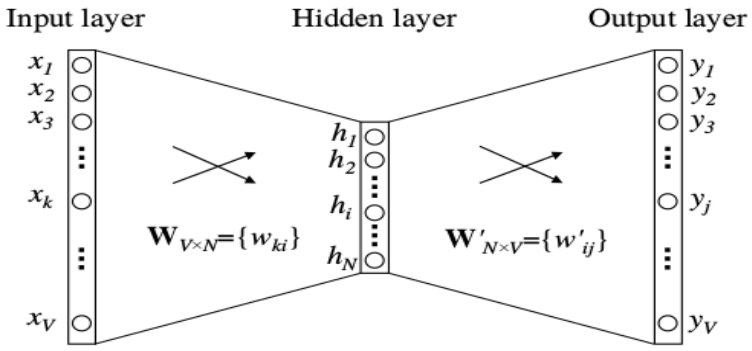

Figure 1: A simple CBOW model with only one word in the context. Image from Rong.

Rong in the paper Rong, the model is composed of a single hidden layer and the network is fully connected Fig(1). In this setting, the vocabulary size is V , and the hidden layer size is N. The input is a one-hot encoded vector therefore only one out of V units, $\{x_1,..,x_V\}$, will be 1, and all other will have value 0.

The weight matrix between input and hidden layer is represented by $\mathbf{W}$.

$$h = \mathbf{W}^T x \tag{1}$$

A different weight matrix W' is used from hidden to the output layer. This matrix is of dimension $N \times V$.

$$u_j = \mathbf{v}'^{\mathbf{T}}_{wj}\mathbf{h} \tag{2}$$

where: $\mathbf{v}_{ij}^{'}$ is jth column od the matrix $\mathbf{W}'$ At the end, a soft-max log-linear classifier is applied at the end of the output layer.

$$p(w_j|w_I) = \frac{exp(v_{w_j}^{'T} v_{wI})}{\sum_{j'=1}^{V} exp(v_{w_{j'}}^{'T} v_{wI})} \tag{3}$$

The error is calculated and then the weights are updated using back propagation algorithm.

2.1.2. Skip Gram Model

Skip Gram model is also introduced by Mikolov et al. [2013b]. The goal of Skip Gram model is opposite to that of CBOW model. Here the goal is to predict the context words as output given a word as an input.

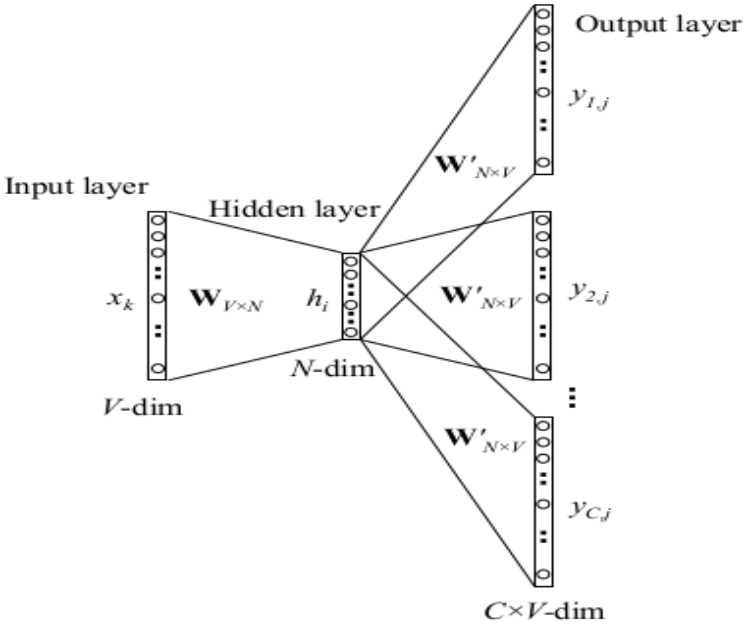

Figure 2: The Skip Gram model. Image from Rong.

In this model precisely, each current word is used as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word. The quality increases if the length of the context is increased because of the fact that if the context(**N** words before and after a particular word), context information is being covered more precisely. This precision comes with an increased training time.

2.2. Word Mover Distance

Word Mover Distance algorithm is proposed by Kusner et al. [2015]. This algorithm make use of Word2Vec model proposed by Thomas Mikolov et al. It makes use of the word embeddings

resulted from Word2Vec Model. The Word2Vec model gives us an embedding matrix where each column represents a word vector of d dim
$\mathbf{X} \in \mathbb{R}^{d \times n}$,
where *n: number of words*
d: dimension of the word embeddings(here it is 300).

The Word Mover Distance is composed of following components which together forms a metric for comparing two documents :

2.2.1. nBOW representation:

Word Mover Distance is a specialized Earth Mover Distance, in which each document is considered as a normalized Bag of Words vector, $\mathbf{d} \in \mathcal{R}^n$. This Bag of word vector representation of each document makes the vectors very sparse. Precisely, if a word, i appears in the document, c_i times then in the vector this word will be represented as $d_i = \frac{c_i}{\sum_{j=1}^{n} c_j}$.

Before converting the document into nBOW representation, preprocessing of the text is done. This includes removing of stop words and various symbols from the document.

2.2.2. Word travel cost:

Each word in the document can be compared to a word in another document. Since one has a vector representation of words generated from Word2Vec, this task become feasible. This also helps in incorporating semantic similarity between individual word pairs from two respective documents which are being compared.

For example, we have two words i and j, then the euclidean distance which can be obtained by using vectors of both the words are represented as *c(i,j)*.

$c(i,j) = \mod^2(x_i - x_j)$

This distance can also be understood as a cost function, if two words are quite different then it will be more expensive to travel from one to another.

2.2.3. Document Distance:

The two components mentioned above, are combined to calculate document distances. It is based on Earth Mover distance which indeed helps in calculating the distance between two probability distribution. It can also be seen as a minimal cost that needs to be paid for transforming one distribution into another. This incorporate earth to be moved from one distribution to another and cost in moving the same.

In Word Mover Distance, the distance between two words *c(i,j)* act as cost for the movement and nBOW of two documents **d and d′** represent two distribution to be compared.

Implicitly, each word i in document $\mathbf{d}$ is transformed to word j in document $\mathbf{d}'$. Let $\mathbf{T} \in \mathcal{R}^{n \times n}$, is a matrix where each value $\mathbf{T}_{ij} \geq 0$, how much word i travels to word j.

To transform document $\mathbf{d}$ into document $\mathbf{d}'$, two constraints need to be fulfilled. These are as follows:

$$\begin{aligned} &\min_{\forall T \geq 0} T_{ij} c(i,j) \\ \text{subject to} : &\sum_{j=1}^{n} T_{ij} = d_i \forall i \in 1....n \\ &\sum_{i=1}^{n} T_{ij} = d'_j \forall j \in 1....n \end{aligned} \tag{4}$$

This is a special case of Earth mover's distance. It can be classified as a transportation problem. The implementation of the Earth Mover Distance we use, was implemented in Python by Ofir Pele and Micahael Werman as cited in Pele and Werman [2008] Pele and Werman [2009].

2.2.4. Optimization techniques:

The Word Mover Distance solves transportation problem implicitly. It fulfills the two constraints while solving the linear optimization problem of Word Mover Distance. Since the complexity of the algorithm is $O(p^3 log p)$ where p: number of unique words in the documentsPele and Werman [2009]. Since the complexity is cubic, Kusner et al. [2015] proposed three optimization techniques.

1. Word Centroid Distance:
Each document can be seen as a vector calculated from the weighted average word. This is very fast to compute using matrix operations. The complexity of calculating Word Centroid Distance becomes O(dp).

This is a good approximation to tell the nearest neighbor search about the promising candidates and helps in significantly increasing the Word Mover Distance search for nearest neighbor.

2. Relaxed Word Mover Distance:
It is a second approach in which the Word Mover Distance is calculated by relaxing one of the constraints one at a time. It gives a better solution than Word Centroid Distance as it gives a tighter bound solution as compared to the former.

When one of the constraints is removed, the optimization becomes:

$$\begin{aligned} &\min_{\forall T \geq 0} T_{ij} c(i,j) \\ \text{subject to} : &\sum_{j=1}^{n} T_{ij} = d_i \forall i \in 1....n \end{aligned} \tag{5}$$

The other optimization will be as follows, when the other constraint is removed, leaving the first one:

$$\begin{aligned} &\min_{\forall T \geq 0} T_{ij} c(i,j) \\ \text{subject to} : &\sum_{i=1}^{n} T_{ij} = d_j \forall j \in 1....n \end{aligned} \tag{6}$$

This can also seen as moving all the probability mass of one word in one document to the nearest word in the second document. The $\mathbf{T}^*$, an optimal matrix will be:

$$T^*_{ij} = \begin{cases} d_i, & \text{if } j = argmin_j c(i,j) \\ 0, & \text{otherwise} \end{cases}$$

This solution is faster than Word Mover Distance as the solution requires only finding $j^* = argmin_j c(i,j)$.
This boils down to comparing the euclidean distances.

This procedure is repeated again with removing the second constraint and applying the first one. The two solutions from solving equations (5) and (6) will be l_1**(d,d′)** and l_2**(d,d′)**. An even tighter bound can be found by taking the maximum of the two. The solution, referred as Relaxed Word Mover Distance will be given by:
l_r**(d,d′)** = max(l_1**(d,d′)**,l_2**(d,d′)**)

3. Pre-fetch and Prune:
In this optimization technique, the search for k-nearest neighbors can be computationally sped up by using the two lower bounds. This makes use of both Word Centroid Distance and Relaxed Word Mover Distance.

The documents are arranged in increasing order of Word Centroid Distance to the query distance. For the first k nearest neighbors, Word Mover Distance is calculated. The rest are traversed and Relaxed Word Mover Distance is calculated for the rest of the documents. If the lower bound of Relaxed Word Mover Distance exceeds the current k^{th} closest distance, then these documents can be pruned. If not, then Word Mover Distance for the document is calculated and if necessary then the k-nearest neighbors are updated.

3. Approach Followed

3.1. Word2Vec

As mentioned earlier, the first task is to build a Word2Vec model using Text8 corpus. This will be compared with pre-trained Google model. This is important because this will help in determining which corpora is better for the task. In Table(1), we can see the differences in both models. We determine the difference on four criterion: Vocabulary size, the number of words used in training, the similarity between two words and accuracy. We cab see that the Google model attains more accuracy than the text8 model. The accuracy of Google model is accountable to the bigger vocabulary size used by Google to train it. When we run the similarity test between words for example: 'address' and 'speak', the Google gives more accurate result.
One interesting fact to notice from the Table(1) is although for the test case of Capital-World, data set is larger for Text8, the accuracy is greater for Google model. This is due to the fact that Text8 has text in lower case. So the context in which a word is used can be different. It is therefor to check and perform the text pre-processing accurately.

Difference In Different Text Corpus		
Criteria	Text8 Corpus Model	Google NEWS Model
Vocab size	71290	3 million
Words used in training	16718843	100 billion
model.similarity('president','obama')	0.2872	0.31
model.similarity('address','speak')	0.10	0.34
capital-world Accuracy:	58.33(847 / 1452)	77.87 (394 / 506)
currency Accuracy:	23.51 % (63 / 268)	35.15 % (161 / 458)

Table 1: Comparison of Word2Vec model generated from Text8 and Google data

Since the Google pre-trained model is more accurate than the text8 corpus, we decided to use it to generate vectors for the Word Mover Distance Algorithm.

We also investigated the vector dimensionality used in the algorithm and why euclidean distances between **vectors of the words** work in the case of high dimensional cases. As stated in Aggrawal et al. [2001] and Bauckhage [2015], we understand that in higher dimensions, euclidean distance does not work as intended, this is also known ***Curse of dimensionality***. Precisely, in higher dimensions, the minimum, maximum and the mean distance become equal.
But we observe that the vectors(dimension 1000) of the words gives similar results as compared to the vectors in lower dimension(dimension 300). For example, the distance between two similar words which are usually used in the same context will have smaller distance in the lower dimension(100dim). This is even followed when dimensionality is increased to 1000dim. This might be due to the fact that the word vectors even in 1000 dim, are not sparse. Hence they still follow the distance metrics even in the higher dimension.

An example is explained in Table(2), where we took two words usually used in the same context and other which is not used in the same context. The distance between similar words is smaller as compared to different words. If the curse of dimensionality was working then the distances would have more or less be equal. Hence curse of dimensionality is not occurring in 1000dim.

Words	1000 dim	500 dim
Girl and Boy	18.19	14.15
Girl and Child	28.72	25.41
Boy and Neither	34.55	33.11

Table 2: Euclidean distance In Vectors of 1000 and 500 dim

3.2. Word Mover Distance

The second step is to implement Word Mover Distance. When implementing this algorithm, we used Google pre-trained model which generated vectors of 300 dim. We also used 20NEWS

GROUP DATA SET. This data set consists of 18000 newsgroups posts on 20 topics split in two subsets. Preprocessing of all documents is also required and thus NLTK package is used to remove the stop words. Using the algorithm stated in section 2.2, Word Mover Distance between a query document and the documents in the set is calculated.

For instance, when we have a query document from an entirely different source which is based on Motorcycle, the k-nearest documents from the set of this document are mostly from the Motorcycle category.

We also implement the three optimization techniques proposed by Matt Kusner et al. We compared the time taken by each of the optimization technique with the Word Mover Distance. For making the comparison we again took the set of 20NEWS GROUP DATA SET and a new data document from a new source. Then using each technique calculated the K-Nearest neighbor, here k=30.

Technique	Time taken (minutes)
Word Mover Distance	276.25
Word Mover Distance with Word Centroid Distance	8
Relaxed Word Mover Distance	90.55
Prefetch and prune	80.25

Table 3: Time taken by different techniques:

The Table(3), shows the different amount of time taken by different versions of optimization techniques. The Word Mover Distance algorithm takes maximum time as it calculates the distance of every document with the query document. The centroid distance is an approximation,which is just calculating the centroid of the documents, hence it is the fastest among other optimization techniques. Relaxed Word Mover Distance and Prefetch and Prune techniques are approximately three times faster.

3.3. Generating Summaries

Generating summaries is one of the most complex tasks in this research field. We propose two approaches to generate summaries of a set of documents. Both of these approaches are based on Word2Vec and Word Mover distance algorithm. Our approaches are motivated by the work of Zhu et al. [2013] and Gross et al. [2013].

Gross et al. [2013], proposed an unsupervised approach for generating summaries known as **Association Mixture Text Summarization**. This approach selects sentences from the document in a manner that they cover all the relevant information. They propose two characters for such sentences, *the relative association between words, novel associations between sentences in the document.* Thus the summarization task can be done in two steps:

1. Computation of document specific associations. 2. Selection of sentences with strong word associations.

Zhu et al. [2013], also proposed a relational learning to rank approach. A ranking function is defined, which will compute a rank for sentences based on content and relation among the

sentences. In this way, one can not only choose the best sentences which not only cover the content but also avoid redundancy using relation among all the sentences.

In this report, we propose an approach, which generates summaries based on the general idea of the documents. We extract the general idea using Word Mover distance between different sets of sentences explained in Section 3.3.1 and 3.3.2. Based on the general idea set, we further refine and extend this set to exclude redundancy and include left out details. At the end, this will be considered as the summary of the documents.

3.3.1. Summary using General idea set which is generated from the whole text document

According to **Harvard writing center**, it is highly probable that we can find the main idea in the first few sentences of the first paragraph or in the last lines of the text document. In the first approach, we assume that the general idea of the given text can be understood through :
1. First few sentences of the paragraphs
2. Last sentences of the document.
3. The most frequent words used in the document.

To extract the main idea from the text, we use these three metrics.The set of sentences selected using these three metrics are represented as **S**. Using Word Mover Distance, we select the most similar sentences to the set **S**. Using distance matrix $\mathbf{D} \in \mathcal{R}^{U \times S}$ makes computation faster. All these sentences are combined to form the general idea set represented as **G**.

Now using the general idea set we will extract the summary. sentences and all the sentences in the text documents are **U**. We build a distance matrix $\mathbf{D} \in \mathcal{R}^{U \times G}$. Each entry in this matrix represents the Word Mover Distance between each sentence in general idea set and each sentence in the all sentence set. To ensure that we have not missed any sentences of any detail, we find the least similar sentence to each of the sentences in the general idea set and also to exclude redundancy we compare this least similar sentence to rest of the sentences in the general idea set for most similarity. If the two sentences have a threshold distance then we add the sentence to summary set besides general idea sentences otherwise, we ignore the sentence.

The resulting set will form summary. The threshold we choose is 0.8. This can be varied and the summary quality will change with this threshold.

3.3.2. Summary using General idea set which is generated from one paragraph at a time:

The second approach also make use of Word Mover Distance algorithm. Here instead of trying to extract the general idea from the whole text document or all documents, we try to find the general idea for each paragraph.

For each paragraph in the text document, we try to find the most similar sentence/sentences to all other sentences for that paragraph. The similarity is calculated using Word Mover Distance between the two sentences. When done for all the paragraphs, we get a set of sentences, represented by **G**. This set may have redundant sentences. A distance matrix $\mathbf{D} \in \mathcal{R}^{G \times G}$ is formed which represents distances between every sentence present in this set. If the Word Mover Distance between any of the two sentences is below threshold = 0.8, then for those two sentences, we determine which one of the two has the maximum distance from the rest of the sentences

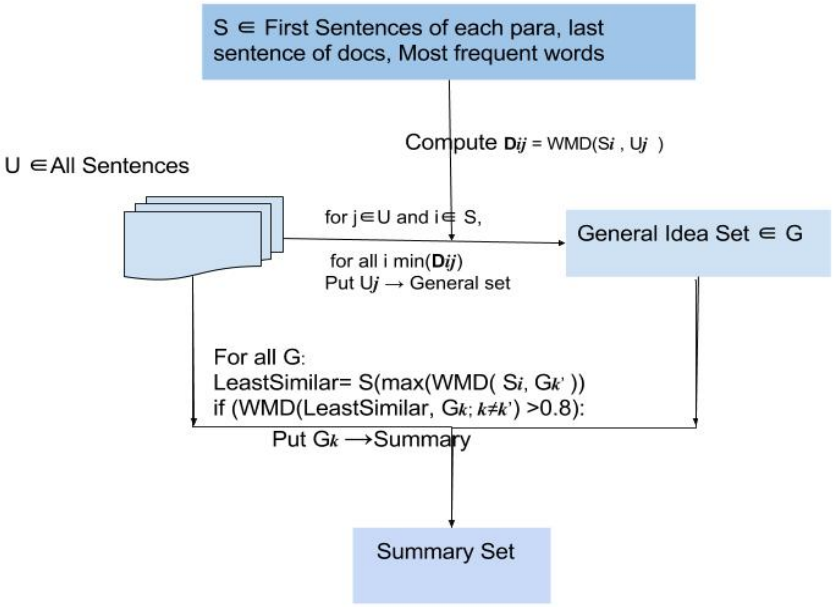

Figure 3: Summary using General idea set which is generated from the whole text document.

in the set. The one with the maximum distance is selected and the other is thrown from the set. This process is repeated for all the sentences. This will represent the summary of the document/documents.

4. Empirical Evaluation

We took three random documents related to a topic. Using both the approaches to generate summary, we are getting the following results. The example documents contains text about Water on Mars. The source of the text documents being :

http://www.businessinsider.de/nasa-liquid-water-mars-2016-9?r=UK&IR=T

http://www.space.com/34095-mars-lakes-suggest-habitable-longer-than-thought.html

http://www.voanews.com/a/mht-mars-may-have-had-water-more-recently-nasa/3512535.html

The text documents are pre-processed for summarization. Note: For better readability, we manually arranged the sentences of the summary.

4.1. Summary using General idea set which is generated from the whole text document:

Summary of the text documents:

The length of the entire text to be summarized is 53 sentences. Using this approach we summaries the text to 7 sentences.

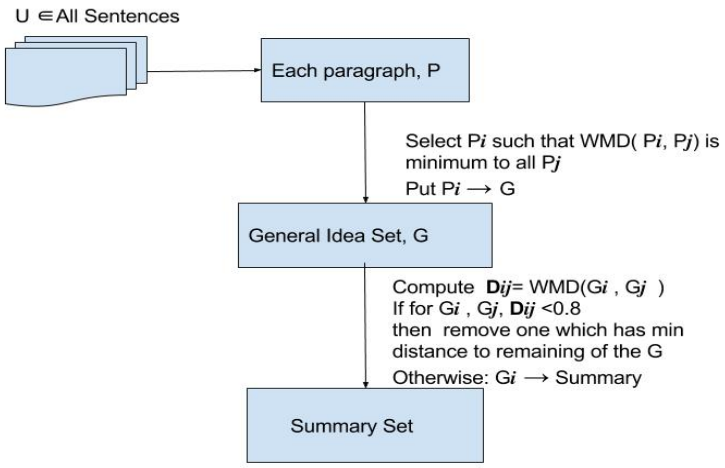

Figure 4: Summary using General idea set which is generated from one paragraph at a time.

Mars may have been able to support life for much longer than scientists had thought. "This paper presents evidence for episodes of water modifying the surface on early mars for possibly several hundred million years later than previously thought, with some implication that the water was replaced by snow, not rain," Mars reconnaissance orbiter (MRO) project scientist rich zurek, of NASA's jet propulsion laboratory in Pasadena, California, said in a statement. Using data from NASAs mars reconnaissance orbiter, researchers say lakes and streams appeared on marks a billion years after a well-documented, earlier era of wet conditions on ancient mars". "We discovered valleys that carried water into lake basins," said Sharon Wilson of the Smithsonian institution, Washington, and the university of Virginia, Charlottesvillee. "The rate at which water flowed through these valleys is consistent with runoff from melting snow," Wilson says. "The team has estimated that heart lake held about 670 cubic miles of water (2,790 cubic kilometers) - more than in lake Ontario, one of north America's great lakes".

4.2. Summary using General idea set which is generated from one paragraph at a time:

Summary of the text documents:
The length of the entire text to be summarized is 53 sentences. Using this approach we summaries the text to 4 sentences.

Using data from NASAs mars reconnaissance orbiter, researchers say lakes and streams appeared on marks a billion years after a well-documented, earlier era of wet conditions on ancient mars. "we discovered valleys that carried water into lake basins," said Sharon Wilson of the Smithsonian institution, Washington, and the university of Virginia, Charlottesvillee. The team has estimated that heart lake held about 670 cubic miles of water (2,790 cubic kilometers) - more than in lake Ontario, one of north America's great lakes. They have simple drainage

patterns and did not form deep or complex systems like the ancient valley networks from early mars." the researchers say they've also found evidence of similar valleys and lakes elsewhere on mars, both north and south of the equator, suggesting that these wet regions were widespread, and not just a regional anomaly. If a valley did cut into such an apron, water was flowing after the crater-creating impact occurred.) observations by MRO, NASA's curiosity rover and other missions had already found strong evidence of lakes, streams and other bodies of liquid surface water in mars' more ancient past 3.7 billion years or so ago.

4.3. Evaluation of results

Both the approaches although have different sentences from the text, are similar in context. They give a general idea about what is being talked about in the documents. One major drawback of these approaches is that we are picking the sentences from the text, it is not generated by the algorithm. Another drawback is that the text is not in a flow, since it selects sentences which can be from any part of the document.

5. Conclusion

In this report, we have a glimpse of Word2Vec and Word Mover Distance algorithm. Different models of Word2Vec have been compared and it is seen that for a better model, more training data is always better. In Word Mover Distance algorithm, we explore all the optimizations proposed by Matt Kusner et al. Both of these algorithms are used to generate summaries of one or more than one text documents. We used two approaches to generate summaries. Both approaches first tries to extract general idea from the documents using Word Mover Distance then refine and extend the general idea set with missed details. The first approach tries to get general idea from the whole document whereas the second goes paragraph wise.

Though the summaries generated from both need improvement as they lack flow of context and are composed of sentences from the text, we can nevertheless use Word Mover Distance and Word2Vec to get a general idea and then summarize the text.

Appendices

Some more examples of the summary generated by the two approaches: Note: For better readability, we manually arranged the sentences of the summary.

A. Example1 : Conservation of Sea turtles

The source is again web source about Conservation of Sea Turtles:
1. http://wenku.baidu.com/view/23667425482fb4daa58d4b46.html
2.http://conserveturtles.org/turtleblog/blog/2016/07/26/tour-de-turtles-competitor-esperanza-spotted-nesting-in-mexico/

3. http://www.conserveturtles.org/seaturtleinformation.php?page=conservation

A.1. Summary using General idea set which is generated from the whole text document

Original text length : 71 sentences
Summary length: 6 sentences
In many states where sea turtles nest, state laws have been passed to protect the species. The threats facing sea turtles are numerous and, for the most part, humans are the problem. For those of us trying to protect sea turtles, it is a mixed blessing that so many threats are human-caused. The intrusion of tourists into these places make it difficult for the turtles to lay their eggs. The indiscriminate collection of turtles eggs on the beaches is no more allowed. One of the benefits of using satellite transmitters to track sea turtles is that it helps us determine turtles nesting site fidelity.

A.2. Summary using General idea set which is generated from one paragraph at a time

Original text length : 71 sentences
Summary length: 5 sentences
In many states where sea turtles nest, state laws have been passed to protect the species. For those of us trying to protect sea turtles, it is a mixed blessing that so many threats are human-caused. The indiscriminate collection of turtles eggs on the beaches is no more allowed. The intrusion of tourists into these places make it difficult for the turtles to lay their eggs. Pollution of the sea has also reduced the number of turtles

References

Charu C. Aggrawal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behaviour of distance metrics in higher dimensional space. 2001.

Prof. Dr. Christian Bauckhage. Pattern recognition. Pattern Recognition Lecture Winter Semester, 2015.

Oskar Gross, Antonine Doucet, and Hannu Toivonen. Document summarization based on word associations. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, 2013.

M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. volume abs/1301.3781, 2013a. URL `http://arxiv.org/abs/1301.3781`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. volume abs/1310.4546, 2013b. URL http://arxiv.org/abs/1310.4546.

Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In *Computer Vision–ECCV 2008*, pages 495–508. Springer, October 2008.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*, pages 460–467. IEEE, September 2009.

Xin Rong. word2vec parameter learning explained.

Yadong Zhu, Yanyan Lan, Jiafeng Guo, Pan Du, and Xueqi Cheng. A novel relational learning-to-rank approach for topic-focused multi-document summarization. 2013.