TextRank Based Extractive English Summarizer

Yunxiao Song

New York University, College of Arts and Science Department of Computer Science

ys2674@nyu.edu

May 2019

Abstract

English Summarizer is an innovative application of natural language processing. In this paper, I introduce a designed English Summarizer System based on TextRank. This system takes any English text as input and extracts sentences with highest score based on their TextRank. It aims at identifying sentences that are more representative for the given text (Mihalcea and Tarau) and extracts sentences that have the best connectivity with the other individual sentences in the context of the sentence. The efficiency of TextRank extraction is analyzed based on ROUGE scores which is based on Chin-Yew Lin's research on testing the recall of automatic summarizer.

1 Introduction

Suppose ranking sentences can be compared with ranking the web page. A summarizer can therefore be realized by adopting close concepts to a search engine that searches and displays the most important sentences from the text input. In terms of workflow, this system has two separated parts. After retrieving the attributes of each English Word, I am going to construct a weighted graph for input text sentences, in which the edge represents sentences' connection as imagined "webpage links". Then I dynamically update the weights of the vertices using a predefined function. The system would eventually find the sentences of highest rank.

The first part is to build a word embedding model trained using Word2Vec as model and Wikipedia as a training corpus. Since ranking the sentences should be based on identifying the attributes that best represents the sentences, based on the design of my system, I choose to perform word embeddings for tokens in the context of Word2Vec. After the word-embedding model is trained with the entire English Wikipedia database, I am going to construct the TextRank of the input text.

TextRank is simply stated, the rank of vertices in which a weighted graph is constructed with edges representing the interconnection between sentences. Every sentence will receive an initial score which represents its significance in the context. All scores together form a score vector, which is used for comparison later. Then, the system performs an in-text training which updates score vectors round by round. The difference of score vectors, which is calculated by the distance, will resultantly converge to zero (My system takes the difference less than 10^{-6} as zero). Finally, the system will output a user-defined ratio of sentences that is indicative of a x percent of most important sentence

2 Prepare of Training Corpus

Wikipedia English is written in English with a broad coverage of vocabularies in all realms. Since my system aims to give a summary for texts in any context, a good coverage of English words is very important, therefore, Wikipedia serves as an ideal open-source training corpus for the word embedding subsystem.

The xml file of English Wikipedia source code can be found on the Wikimedia Dump website. This system applies the Wikipedia extractor from Medialab to extract valid written English text from html source code. The extracted file still contains "<doc>" tags, which was further cleaned using Regexp.py file in python. In order to exclude the influence of stop words, because the stop words are not representative of the sentence as a whole in terms of sentence featuring. This program also cleans the stops words labeled by

Google SEO Stop Words list. After preprocessing of the Training Corpus, the next step is to introduce Word2Vec into the system.

3 Word2Vec Modeled Word Embedding

Since the training corpus is gigantic in terms of words and sentences, it is impossible for me to embed the words using TF-IDF word embedding because such different sentences to the degree that they appear in wiki databases would result in an unfavorable vector size. Therefore, I am searching for a way of performing word-embedding that gives us precise fixed-dimension vectors that are not too long for all words. Finally, I choose Word2Vec model to be the word embedding model in the system. Word2Vec uses recurrence neural network to perform the word embedding which enables us to represent words using a predefined dimension of vectors (this feature dramatically makes the size of vector shrink). Another reason is, Word2Vec representations are surprisingly good at capturing syntactic and semantic regularities in language material. Each relationship is also characterized by a relation-specific vector offset. This allows vector-oriented reasoning based on the offsets between words (Mikolov, Yih Zweig). This capture of internal features is exactly what the summarizer system needs in terms of information retrieval.

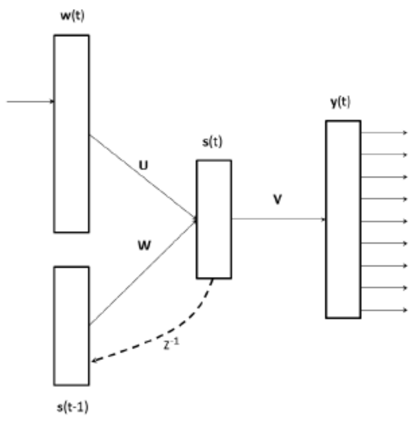

Figure 1. Recurrent Neural Network Model (Mikolov, Yih, Zweig)

The Word2Vec model uses recurrence neural network model which consists of an input layer, a hidden layer with recurrent connections, plus the corresponding weight metrices. The input vector w(t) represents input English word at time t encoded using 1-of-N coding, and the output layer y(t) produces a probability distribution over words. The hidden layer s(t) maintains a representation of the sentence history. The input vector w(t) and the output vector y(t) have dimensionality of the vocabulary. The values in the hidden layer and the output layer are computed as follows:

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)\right) \quad (1)$$

$$\mathbf{y}(t) = g\left(\mathbf{V}\mathbf{s}(t)\right), \quad (2)$$

where

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}. \quad (3)$$

In this framework, the word representations are found in the columns of U, with each column representing a word. The RNN is trained with backpropagation to maximize the data log-likelihood under the model. The model itself has no knowledge of syntax or morphology or semantics. Remarkably, training such a purely lexical model to maximize likelihood will induce word representations with striking syntactic and semantic properties, which are significant for further application of those representations

For this system. I used a python version of Word2Vec from Gensim Research Group. The model is a trained skip-gram Word2Vec model, which embeds each word into a 200-dimension vector, while using 10 windows and 15 iterations with 8 workers in training process. This training mostly demands CPU and RAM usage on features of the task. The computer that I used for this particular research is a Core i7-8550U at 1.8GHz which has 4 cores and 8 threads; therefore, I use 8 workers to fully integrate multithreading into the training process on CPU. The Training approximately took 30 hours to finish. The word2vec_test.py file is used as an interface to test corpus which tests how well the word embedding method works. I chose a word list of 10 words, conducted a small evaluation of this model. The words are: {Apple, language, process, math, graph, computer, calculate, award, significant, president}. For each word, I use the model to search English words with highest cosine similarity using their vectors, and the result is listed in the chart:

Original Word	Similar Word Most Similar	Similar Word Second Similar
Apple	IOS	iPhone
language	spoken	dialect
process	method	involves
math	mathematics	students*
graph	vertices	undirected
computer	software	hardware
calculate	equation	formula*
award	achievement	prize
significant	substantial	significance
president	vice-president	chairman
tiger	gaur	deer
nationality	Citizenship	ethnicity*
glory	Heaven*	god*
bottle	Jenever*	Wine*

Figure 2. system-predicted most similar words to the original word. Predicted by using English Wikipedia Corpus trained skip gram Word2Vec, words that have similarities lower than 0.50 are labeled with *. Similarities are calculated using cosine similarity, pl form of the word is not taken into account.

based on our test, the word2vec model works well on embedding the word into vectors. Word Vectors will be a core attribute that I use to construct the TextRank graph.

4 Construct TextRank Graph

Firstly, I will introduce the concept of TextRank Graph. Let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges $E$where E is a subset of $V \times V$. For a given vertex V_i, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V points to (successors). The score of a vertex is defined as Follows (Brin and Page, 1998):

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

Where d is a damping factor that can be set between 0 and 1, which has the role of integrating into the model of the probability of jumping from a given vertex to another random vertex in the graph. In the Page rank model is the factor *d,* which is usually set to the 0.85 (Brin and Page, 1998). I use the identical d value in my Summarize system.

In my system, the graph is built from any possible natural language texts. In terms of the relation between sentences, we need to include multiple or partial links between unit sentences that are extracted from text. Since those links vary by a lot, It may be useful to indicate and incorporate into the model the strength of the connection between two vertices V_i and V_j as a weight w_{ij} added to the corresponding edge that connects the two vertices. Consequently, Brin and Page's team introduced a new formula for graph-based ranking that takes the edge weights into account when computing the score associated with a vertex in the graph. The modified similar formula from page rank is

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Since Mihalcea and Tarau had proved that the weighted graph would still converge as the unweighted graph with almost identical number of iterations, we will run evaluation until the score of each vertex converges to a value in that limited number of iterations. (Mihalcea and Tarau, 2004)

5 Mapping Text to Graph

The first step of Mapping the text to the graph is preprocessing the text. I firstly split the text into unit sentences, constructing a list of unit sentences. The next step is copying the sentence list and splitting each sentence into tokens, after filtering the stop words in tokens, sentences are passed into the step of constructing the graph.

To make TextRank work on any natural input text, this system has to build a graph that represents the text while interconnecting words or other text entities with meaningful relations. For the input text in my system:

1. *The sentence is identified as the text unit that is best defined as vertices in the graph.*
2. *The relation that connects the sentences is the original similarities between the sentences. And those edges are not directed*
3. *Iterate the graph based ranking algorithm until convergence*
4. *Sort the vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions*

This is how my system tackles the natural text input. As I illustrated, the edges between the sentences is defined to be their similarities, and the similarity between sentence_1 and sentence_2 can be calculated by average cosine similarity:

$$sent1 = \sum_{for\ word\ in\ sent1} model.vec(word)$$

$$sent2 = \sum_{for\ word\ in\ sent2} model.vec(word)$$

$$similarity(sent1, sent2) = \frac{sent1 \cdot sent2}{\sqrt{\|sent1\|^2 \cdot \|sent2\|^2}}$$

Consequently, this similarity between the sentences is the very initial relationship between the sentences. The graph would be iterated until it converges to a small neglectable value.

6 Evaluation Method Introduction

In terms of how to evaluate the performance of the system. I am incorporating ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to test the system outputs.

ROUGE includes measures to automatically determine the quality of a summary by comparing it to other summaries (ideally summaries created by humans). The measure counts the number of overlapping units such as n-gram, word sequences, and word pairs between the computer generated summary to be evaluated and the ideal summaries created by humans. My summarizer uses ROUGE-1 and ROUGE-L to measure and evaluate the performance. (Lin, 2004)

ROUGE-1 refers to the overlap of 1-gram between the system and reference summaries. This is adopted in my evaluation because the ROUGE-1 reveals the similarities of single significant token between the system generated summary and human summaries. This is intuitive because the we normally prefer a candidate summary that is more similar to consensus among reference summaries.

ROUGE-L refers to longest common sequence based statistics. This feature takes the sentence level similarities into account. It is reasonable to expect a summary that is more similar to human summaries to be a good summary. Therefore, the longest common summary is able to serve as a meaningful measurement of the quality.

7 Results

ROUGE evaluation requires high quality human translation, therefore I carefully used Opinosis dataset which concentrates on real human's opinions. The research group of Opinosis collectively organizes human opinions databases and the gold-summary from human. Each opinion text has 5 gold-summary in order to enhance the measurement, those gold-summaries partially rewrote from the original text and may use different word choices, but the sentences are sequenced as the original text. This is the first dataset used to evaluate the system. And based on the properties of the system, I am evaluating the system under two different pre-set ratio of text that are extracted. The first one extracts 3 most important sentences, the second one extracts the sentences that has the TextRank score in the top 0.3 percentile, and this score as a result represents their calculated importance. Therefore, the second extracts the top 0.3 most important sentences based on their importance. Both of the experiments are carried out using the ROUGE package in Java version with all the data properly preset.

It provides immediate alternatives if the route from the online map program was inaccurate or blocked by an obstacle .I've used other GPS units, as well as GPS built into cars and to this day NOTHING beats the accuracy of a Garmin GPS .It got me from point A to point B with 100% accuracy everytime . It has yet to disappoint, getting me everywhere with 100% accuracy. 0 out of 5 stars Honest, accurate review, PLEASE READ ! Aside from that, every destination I've thrown at has been 100% accurate. In closing, this is a fantastic GPS with some very nice features and is very accurate in directions. Plus, I've always heard that there are quirks with any GPS being accurate, having POIs, etc .DESTINATION TIME, , This is pretty accurate too .But, it's always very accurate .The map is pretty accurate and the Point of interest database also is good . Most of the times, this info was very accurate. I've even used it in the pedestrian mode, and it's amazing how accurate it is. ONLY is only accurate when an ad says, Top sirloin steak, ONLY $1 .

Figure 3. the first 20% of the first Opinosis text used in the test, the test corpus is mainly formed by short sentences of opinions, and my summarizer system serves to extract the most representative comments from the test corpus. As we can see, parts of the sentences are informally written.

1. I've even used it in the pedestrian mode, and it's amazing how accurate it is .
2. If your looking for a nice, accurate GPS for not so much money, got with this one .
3. Very Accurate but with one small glitch I found , I'll explain in the CONSThis is a great GPS, it is so easy to use and it is always accurate .

Figure 4. the three sentences extracted using extractive automatic summarizer, Figure 4. And Figure 3. Serves as an example of input and output for the system.

I chose the first five texts from that database and use the system to extract the 3 most important sentences. the system output is compared with the 2 gold-summaries from human. Since the human written summary is very precise, for example, it summarizes 200 sentences to approximate 3 sentences of 10 words each. This situation naturally makes the precision of extractive summary low and meaningless. Therefore, the measurement would mainly be recall oriented, and we can easily find that the worst possible recall would be less than 5 percent if we randomly chose 3 sentences as our result of extraction. Moreover, the reason that we chose 3 sentences as the quantity of our automatic generated summary is that the length of human gold-summary is usually 2 or 3 short sentences. Therefore, the result which is shown in the chart below is acceptable.

ROUGE---type	Task-Index	Average recall	Average f-score	Extract Ratio
ROUGE-L	1	0.33	0.21	0.047
ROUGE-1	1	0.33	0.18	0.047
ROUGE-L	2	0.31	0.25	0.035
ROUGE-1	2	0.38	0.27	0.035
ROUGE-L	3	0.25	0.12	0.036
ROUGE-1	3	0.25	0.10	0.036
ROUGE-L	4	0.38	0.27	0.05
ROUGE-1	4	0.39	0.27	0.05
ROUGE-L	5	0.39	0.20	0.0098
ROUGE-1	5	0.42	0.18	0.0098

Figure 5. The ROUGE score for the extractive summarizer. Extract ratio shows what ratio of original text is extracted by the summarizer. The performance of the summarizer is overwhelming in terms of having a recall score with that small extract ratio.

After the 3-sentences extraction evaluation is completed, how the system works in a fixed extract ratio (30%) is evaluated with average recall and average f-score, which is fixed to be 0.3, with all the other evaluation settings the same, data is displayed in the chart below. There is no extract ratio shown, because the extract ratio is pre-set to 30% with less than 1 percent difference in ratio

ROUGE---type	Task-Index	Average Recall	Average f-score
ROUGE-L	1	0.63	0.085
ROUGE-1	1	0.63	0.060
ROUGE-L	2	0.58	0.085
ROUGE-1	2	0.60	0.019
ROUGE-L	3	0.75	0.064
ROUGE-1	3	0.75	0.037
ROUGE-L	4	0.62	0.091
ROUGE-1	4	0.82	0.082
ROUGE-L	5	0.85	0.031
ROUGE-1	5	0.85	0.015

Figure 6. the ROUGE based score for extractive summarizer when extracting a fixed ratio (30%) of the sentences from the original text. As we can see, the average recall ranges from 0.58 and 0.85, which is very high considering this system only extracts 30% of the sentences from the text

.

Since the human gold-summary is written by human experts with reorganized English and differed word choices, the 5 gold-summary for each long input file varies with each other, therefore, as a comparison, I am doing the

ROUGE score of 4 individual gold-summary of same text as the reference, and 1 other different gold-summary of the same text as pseudo-system-output. Doing the ROUGE between human gold-summaries gives us an idea of the ideal score for an English summarizer. Here is the chart that roughly estimates what will be the ideal result from an optimized English summarizer.

ROUGE---type	Task-Index	Average recall	Average f-score
ROUGE-L	1	0.16	0.15
ROUGE-1	1	0.16	0.15
ROUGE-L	2	0.20	0.22
ROUGE-1	2	0.19	0.21
ROUGE-L	3	0.42	0.46
ROUGE-1	3	0.44	0.48
ROUGE-L	4	0.46	0.55
ROUGE-1	4	0.44	0.54
ROUGE-L	5	0.66	0.63
ROUGE-1	5	0.68	0.66

Figure 7. the ROUGE score of 1 gold-summary as pseudo-system output, using 4 other gold-summary as references. This chart proves that even if the summary comes from human experts, the data still exists a boundary in recall and f-score when using the ROUGE evaluation. Based on observation, an average recall of more than 0.38 is excellent for an automatic summarizer. And a f-score of 0.40 is perfect for an automatic summarizer. Comparing Figure 5. With Figure 3. we find the conclusion that the summarizer works well

As a conclusion, the system achieves a pretty good score based on our test. For 3-sentences extraction task, the extractor reached a recall score that is way higher than my expectation. Because as the Figure 7 shows, the upper bound of summary recall rate for 3-sentence extraction itself is low. Meanwhile, from my perspective the sentences extracted is representative when compare them with input comments and expert summaries. However, the discussion on evaluation also reflects the fact that the criteria to judge an extractive summarizer system is not clear. With the increase of the text size, it becomes harder and harder for human to reach a consensus on what is the best summary. Therefore, with the criteria of evaluating summary quality is not statistically defined, it is hard to judge the performance of automatic summarizer.

8 Discussion:

Firstly, I find the evaluation of summary systems hard to carry out because the definition of a good summary is always vague. As the data of Figure7. shows, even the summaries of the same text from human experts would vary a lot. Therefore, in terms of evaluating the quality we need to figure out what is considered to be an informative and accurate summary. A more precise definition of summary and a more precise measurement can be the basis of further research in the field of automatic extractive text summarizer.

My system defines sentences that share more similarities with other sentences to be significant sentences. But this definition will not work in some situations. Although I had all the stop words removed in my graph, there could still exist sentences that share huge similarities with other sentences, but are not representative at all. A more representative formula that can represent the relationships between the sentences will be the next stage for such research. For example, we can combine the sentence similarities with other attributes, do regressions on it and evaluate the attributes to see whether it improves the performance or not.

Overall, I found an important feature while doing research with my system. My system can be adapted to a summarizer of any sort of language. For example, if we want to adopt the system to summarize the text in French, we just need to substitute the training corpus, making sure that the decoding method works. So long as the Latin text can be chopped into sentences, this system would be able to apply text rank on it. For a language like Chinese that by nature does not have any spaces in it, we need a tokenizer in order to get text with well-separated tokens. As a conclusion, this summarizer system is versatile in terms of adapting different languages.

9 References

Chin-Yew Lin, *ROUGE A Package for Automatic Evaluation of Summaries*, University of Southern California, 2004

Daniel S. Leite1, Lucia H. M. Rino1, Thiago A. S. Pardo, Maria das Graças V. Nunes, *Extractive Automatic Summarization: Does more linguistic knowledge make a difference?* 2007 Association for Computational Linguistics

gensim, *Models on word2vec-word Embeddings*, https://radimrehurek.com/gensim/models/word2vec.html

Medialab, *Wikipedia Extractor*, http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

Rada Mihalcea and Raul Tarau, *TextRank: Bringing Order into Texts*, University of North Texas, 2004

S. Brin and L. Page. *The anatomy of a large-scale hypertextual*, 1998

Tomas Mikolov, Wen-tau Yih, Geoffrey Zweig, Linguistic *Regularities in Continuous Space Word Representations*, Microsoft Research, 2012