Trend Analysis Using Contextualized Newspaper Articles

Jungyeon Koh
Chickmagalur Basavaraja

Contents

Abstract

We implemented a contextualized word representation using ELMo which represents not only syntactic word uses but also its linguistic contexts. ELMo is a distinct embedding method from conventional ones in that the whole input sentence is allocated to a single vector. We then conducted a trend analysis based on this pre-trained model using GNG algorithm. GNG is an optimal method to represent topological structure of given data set by flexibly adjusting the data units. The result showed a cluster of news articles assorted by the similarity among them.

1 Introduction

Word representation, pre-trained by a number of text corpora is an essential part in many neural language models. In order to attain a quality of result, we conducted "deeply contextualized" word representation using ELMo. It is a state-of-the-art method which notably improves the language understanding problems.

ELMo (Embeddings from Language Models) is a distinct embedding method from conventional ones in that the whole input sentence is allocated to a single vector. It uses vectors, pre-trained with a coupled language model (LM), which is the reason why it is called ELMo. ELMo is a function of every internal layers stacked over each other to develop a linear formulation.Such internal states allow a very deep word representation.

With the input signals refined by ELMo, GNG (Growing Neural Gas) would conduct a topology learning which locates an optimal topological arrangement representing a given data distribution. Following the spirit of the Hebbian rule, GNG produce a well-fitted topological structure for a given input space by incrementally adding a modified position vector to the network toward the input signals.

Trend refers to a universal topic that affects human life among other things, which change uncertainly due to various factors such as time continuity. Trend analysis is roughly a method of collecting widely distributed information and catching a trendy pattern. The significance of trend analysis is that it makes it easier to predict the future by gaining a high-level picture of current phenomena. In our simply-planned project, we cannot expect such kind of result, but we would like to evaluate how accurately our algorithms have predicted the past trends.

Trend analysis upon text inputs can track how the word usage have changed upon time, which is formally called diachronic analysis. It could spot neologisms and archaisms, which are newly-made words and archaic words rarely used other than artistic or literary style, respectively. Google provides customers with Google Trends to search for how particular trends of words have been in its search portal. Similarly, the topological

structure by GNG would implicitly indicate a gradual difference of trends for given time horizons.

2 Related Work

2.1 ELMo

Using pre-trained word vector is one of a well-known natural language processing methods regarding its capability of capturing the syntactic and semantic context of a number of words. However, before the introduction of ELMo, such approaches were only able to track a single context for each word which was totally independent from others.

Some recent works have continuously sought for context-dependent word representation which could deeply reflect the context of a whole sentence. For instance, **context2vec** used bi-LSTM (bidirectional Long Short-Term Memory) to encode the pivot word according to its neighbor word nodes. Furthermore, previous work has discovered a 2-layer LSTM encoder can efficiently predict the POS (part-of-speech tags).

Such approaches led to the introduction of ELMo (Embeddings from Language Models), which successfully represent the contextualized sentence embedding.

2.2 GNG

GNG was initially devised by Bernd Fritzke in 1995. A number of experiments since then has been conducted using the GNG algorithm. GNG is widely used due to its adaptability to any kinds of input distribution without using some elaborately pre-determined parameters.

However, GNG has a significant weakness. It does create a topological structure incrementally, but cannot readily adapt to a rapidly changing input distribution. Such downside led the same author, Benrd, to improve his GNG algorithm by introducing a newly-designed algorithms, the GNG-U and SGNG later in 2003. They both are widely discussed nowadays in terms of their high utility in many different domains. GNG-U allows the algorithm to readily track non-stationary input distribution by re-distributing some rarely used nodes, in other words, nodes which contribute little to the topology, by taking a certain local parameter as a criterion. SGNG is a supervised version of the GNG algorithm using the RBF (Radial Basis Function) network. RBF is a kind of an ANN (Artificial Neural Network) which is a fully interconnected feed-forward network with only one hidden layer.

We used a GNG-U version to successfully make a cluster of the articles.

3 Theory

3.1 ELMo: Embeddings from Language Models

The problem with most of the well-known word embedding techniques are their inability to learn the context of the text as a whole. This could be easily solved by the sentence embedding. Sentence embedding formulates a representation for each sentence by a series of numbers, or a vector. ELMo (Embeddings from Language Models) is known as one of the best algorithms to convert sentences into certain vectors with an advantage of driving the context of the sentence.

3.1.1 Long Short-Term Memory

The problem with conventional Recurring Neural Network (RNN) is that gradient-descent method shrinks over time as the gradient value becomes too small to contribute much to the learning. Hence, the RNN has short-term memory. In order to overcome such problems, LSTM (Long Short-Term Memory) has been introduced to keep the relevant information in track and drop the other irrelevant ones which are not required for further predictions.The words of sentences are converted into relevant vectors and then are sequentially passed through the inner network in LSTM.

There are a number of operations conducted in the LSTM cell which help to keep or remove the information if necessary. The inner hidden states act as a memory of a neural network, carrying the information from the first step to the last one. Information is added or removed via gates for each step. 3 gates are normally used; input gate, output gate and forget gate. Each of them contains sigmoid activating function respectively, squishing the values into range between 0 and 1.

Forget gate decides whether the information provided should be kept or discarded. The data which is held by the previous cell and the new input signals are passed through its sigmoid function. The function determines the current cell's behaviour by its output; if the output is closer to 1, keep the data, otherwise, if the output is closer to 0, forget the data and discard it. Next, these two values would be passed through the input gate, which updates the state of the current cell. Similarly, the closer the output of its sigmoid function is, the more significant the data is. The input gate also contains tanh activation function which squish the value into range between -1 and 1 in order to regulate the network. The sigmoid output will indicate which information should be kept from the tanh output.

In order to formulate the newly-updated cell state, several calculations are held. First, the previous cell state is multiplied by the forget vector output. Then, it is polarized-added with the input vector output. Such result would be a new cell state with updated

values.

Lastly, the output gate tells which information the next hidden state should be carrying. Hidden state information and current data are passed through the sigmoid function of the output gate, then the result is multiplied with the newly updated cell state. The hidden state would be then formulated by passing it rhough the tanh function. After passing these three gates, the newly updated cell state and hidden state would be carried over to the next time step. Such processes for each inner cell of LSTM involves passing information from the previous hidden state to the next one as it propagates forward.

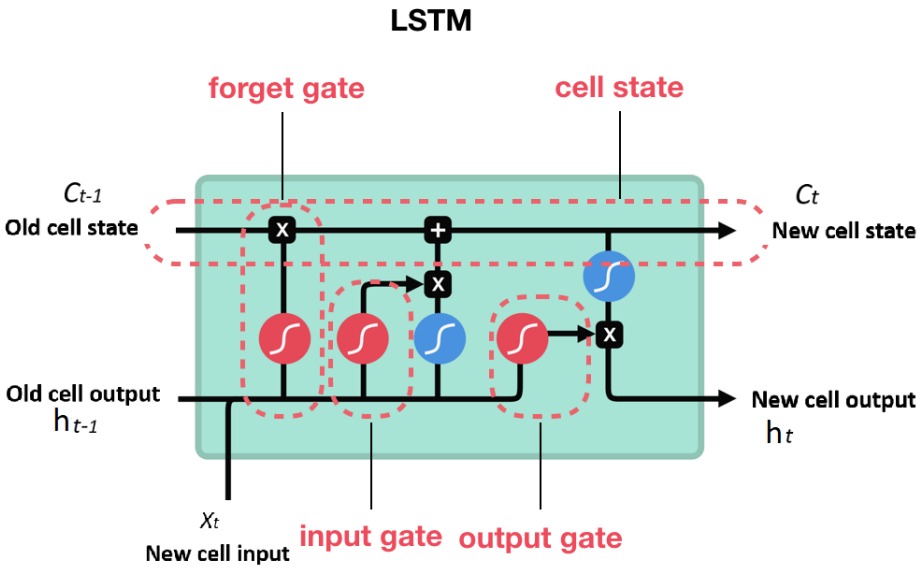

Figure 1: LSTM's internal mechanism

Bi-LSTM is a model which consists of two layers of LSTM, with one in forward-direction and the other one in backward-direction. The forward-directional LSTM keeps the information from the previous actions, and the backward-directional LSTM keeps the information from the next cells.

3.1.2 Embeddings from Language Models

Elmo works in two-layered bidirectional LSTM and it uses character-level convolutional neural network to convert the sentences into raw word vectors. Such vectors would be the input to the BiLSTM and its two layers have two passes respectively, forward-pass and backward-pass. The forward-pass contains the information of the context of the previous word, while backward-pass contains information of the context of the next word. These two passes form an intermediate word vectors and are sent to the next layer

of Bi-LSTM. Such steps would lead to the final representation, which contains weighted sum of every word vector and the intermediate vector.

It is character-based mechanism because the model forms the final representation according to its pre-trained vocabulary corpora. However, what makes it different from other traditional word embedding techniques is that the vectors of the words themselves are functions formulating the text string (sentence) which contains that word.

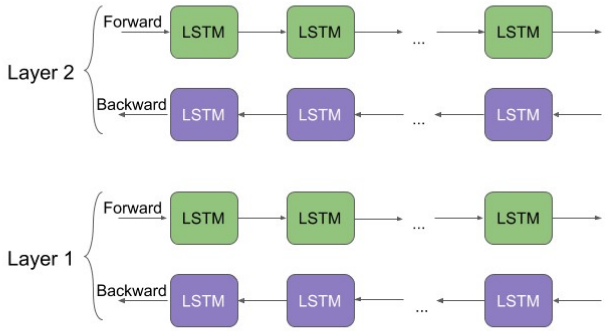

Figure 2: ELMo's two-layered network

3.2 GNG: Growing Neural Gas

3.2.1 Competitive Hebbian Learning and Neural Gas

Topology learning is one of the objectives of a unsupervised learning. Given a certain high-dimensional data distribution $P(\xi)$, it locates a topological arrangement which highly represents the characteristics of the data distribution. For such conducts, "competitive Hebbian learning" (Martinetz, 1993) is commonly used with the "neural gas" (Martinetz and Schulten, 1991) which allows some vector quantization.

The principle of "Competitive Hebbian Learning" (CHL) is:

> For each input signal x, connect the two closest centers (measured by Euclidean distance) by an edge.

Such principle leads to a resulting subgraph which is called the "induced Delaunay traingulation". It depicts the area of the input space where $P(\xi) > 0$. In this sense, it adequately sustain the topology of a orginal data distribution.

However, CHL itself cannot guarantee that every centers can develop any edges since some located in the neglected part can remained as dead units. Martinetz and Schulten have introduced a distinct kind of vector quantization method, called the "neural gas" (NG). The main principle of NG is:

> For each input signal x, adapt the k nearest centers whereby k is decreasing from a large initial to a small final value.

Initially, k value is large enough to cause adaptation for a number of the centers to be shifted toward the input signal. Then k value gradually decays resulting in only a fraction of centers to be adapted by the input signal. However, to accomplish such principle, it is necessary to decide on the total number of adaptation steps and adjust the k value based on it anteriorly.

The combination of CHL and NG is a quite successful manner of topology learning. However, it is quite vague to determine an adequate number of centers in practical applications. The "Growing Neural Gas" (GNG) algorithm overcomes such annoyance and provide several other advantages as well by flexibly manipulating data centers.

3.2.2 Growing Neural Gas Algorithm

The main idea of GNG algorithm is to continuously add new nodes to an original small network by rating local statistical scales, resulting in a finalized topological structure with a fixed dimension (for example, 2D or 3D).

The topological structure is built incrementally by CHL and its dimensionality depends on the characteristics of the input data.

The network which GNG handles consists of:

1. a set A of nodes. Each node $c \in A$ has a correlated reference vector $w_c \in R^n$. w_c indicates the position of node c in the input space.
2. a set N of edges between pairs of units. These edges are not weighted. They only define the topological structure.
3. a number of n-dimensional input data following an unknown probability density function $P(\xi)$.

The complete GNG algorithm is conducted as following:

1. Begin from two units a and b chosen which is located in w_a and w_b randomly chosen from R^n.
2. the input signal ξ in accordance with $P(\xi)$.
3. Find the nearest node s_1 and the second-nearest one s_2.

4. Increment the age of all edges starting from s_1.

5. Store the squared distance between the input signal ξ and the nearest unit w_{s_1} to a local variable:

$$\Delta error(s_1) = \| w_{s_1} - \xi \|^2$$

6. Change the location of s_1 and its direct topological neighbors toward ξ by small fractions, ϵ_b and ϵ_n, respectively:

$$\Delta w_{s_1} = \epsilon_b(\xi - w_{s_1})$$

$$\Delta w_n = \epsilon_n(\xi - w_n) \text{ for all direct topological neighbor } n \text{ of } s_1$$

7. If s_1 and s_2 are linked within an edge, set the age of this edge to zero. If such edge does not exist, create it. (Such conduct is Hebbian in its spirit.)

8. Eliminate edges whose age is larger than a_{max}. If such results in nodes without any emanating edges, remove the nodes as well.

9. If the number of input handled since is an integer multiple of a predefined parameter λ, conduct some additional behavior as follows:

9-1. Settle a new local vector q with the maximum accumulated error.

9-2. Define a new vector r as an average point between q and f whose error variable is the largest among the neighbors of r.

$$w_r = 0.5(w_q + w_f)$$

9-3. Insert a new edge connecting the new vector r with q and f, and remove the existing edge between q and f.

9-4. Reduce the error variables of q and f by multiplying with a constant α. Initialize the error variable of r with the newly set error variable of q.

10. Reduce every local error variables by multiplying with a constant d.

11. If a terminating requirement (e.g., the volume of network or some performance testings) is not accomplished, start from the beginning, Step 1.

The adaptation steps towards every input signals result in general movement of every position vector toward given input space. Such algorithm enables GNG model to quickly learn some essential topological traits of given input's density function, finally constructing a rather well-fitted topological structure.

4 Implementation

We used the json data containing 202,372 records of news articles headlines collected from The Huffington Post. The assorted data was kindly given by Rishabh Misra by his website in Kaggle, `https://www.kaggle.com/rmisra/news-category-dataset`. Each json record contains several attributes: category, headline, authors, link, short description and the date.

A specific example for our data is followed:

{
"category": "ENVIRONMENT",
"headline": "Keystone XL: Time for the Senate to Show Some Courage",
"authors": "Bill McKibben",
"link": "https://www.huffingtonpost.com/entry/keystone-xl-protests_us",
"short_description": "We've been to jail, we've marched on Washington; this week it's pixels and keystrokes. This electronic blitz is an effort to show Congress that there's support out there for doing the right thing.",
"date": "2012-02-11"
}

4.1 Sentence Embedding

The machine learns through numbers which is converted from any given input signals to get an output. Similarly, by word embedding, we are able to convert a given text into computer-readable numbers. Word embedding is a technique of mapping each word into a vector consisting of real numbers in a predefined vector space.

Then how do we get such numbers of a vector such as $b = [0.10.750.6]$?

At first, we considered using word2vec for word embedding, but since it does not represent the whole context of the sentence, we decided not to use it."Context" here implies that the meaning of a word can differ from one sentence to another. For example, let's consider these two sentences: 'He likes to play football.' and 'The play performed by him was good.' In these two sentences, the word 'play' has a different meaning. However, using word2vec cannot represent the difference between those two distinctly. Furthermore, we noticed that there are some sentences which use different words but rather in the same contextual circumstances. In such cases, word embedding is useless.

To solve such incorrect embedding problems, we introduced a new embedding technique called ELMo which handles sentence embedding rather than word embedding.

Also, we decided to use only the headlines rather than the whole simple description of the article to conduct the embedding and further analysis in order to reduce the

complexity of our project. We discovered that including more text to be encoded into numeral vectors would increase the necessary batch size in a large scale, which would obviously lead to a much longer operating time and a wasteful management of the computer memory.

A single headline is transformed into a vector type consisting of a number of dimensions. According to each vector representing the relative location of an single input in its input space, the similarity between articles can be also calculated.

Following is the example for our ELMo encoding.

Sim: 1.0 / best plus size models: who is dominating the industry right now (photos)
[0.19114903 0.07474463 0.04324702 ... 0.12134494 -0.24336293 0.04168314]
Sim: 0.71750003 / the hottest braid in hollywood right now is...
[-0.17134684 0.40277442 0.3884763 ... -0.00901962 0.01219993 0.13390625]
Sim: 0.712546 / this throwback hair trend is huge on pinterest right now
[0.10584313 -0.07349755 0.14238161 ... -0.186909 0.236025 0.07090683]
Sim: 0.7070953 / interior designer michelle workman shares the trends she's tired of and what's hot now (photos)
[0.11854073 0.06395374 -0.02415386 ... -0.19865113 0.38358566 0.21848065]

Following are the plots showing our ELMo implementation. Due to the difficulty to depict n-dimensional grid properly, we only used the first two dimension for a graphic plot.

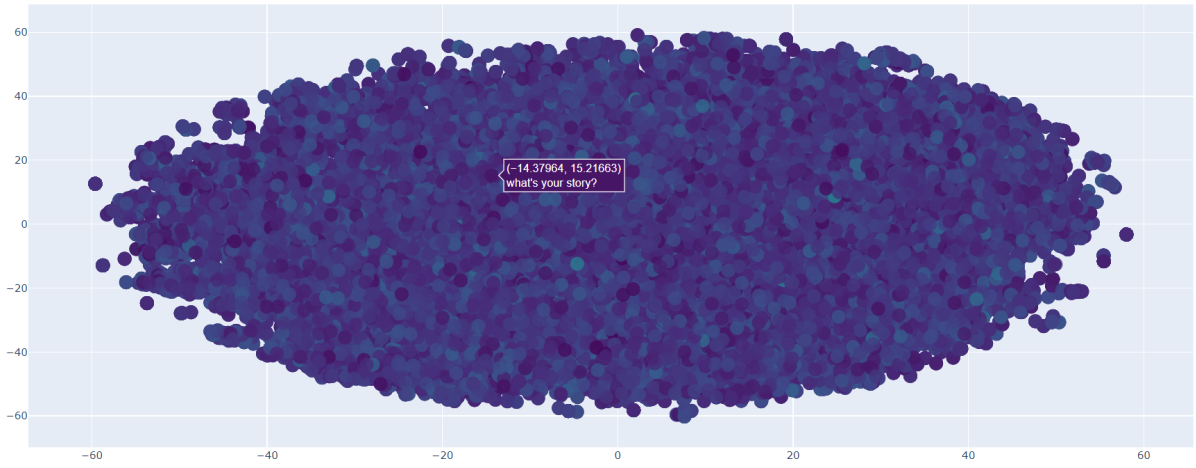

Figure 3: Result of Sentence Encoding For 80k Articles

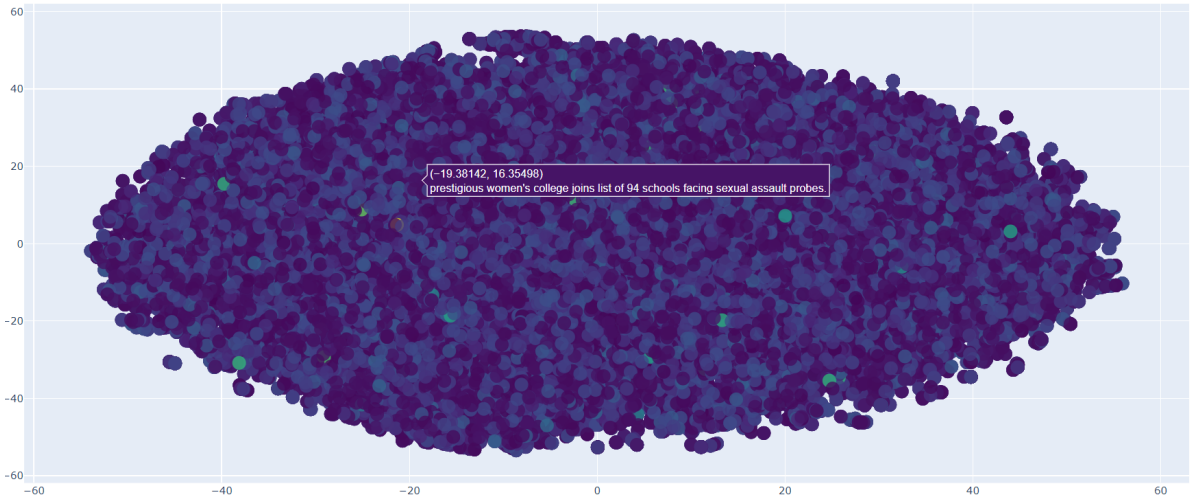

Figure 4: Result of Sentence Encoding For 100k Articles

4.2 Topological Structuring

This project applied an incremental clustering algorithm of Growing Neural Gas (GNG) devised by Bernd Fritzke. The reason why we decided to use GNG other than K-means clustering algorithm or Kohonen's Self Organizing Map (SOM) is to take advantage of its "incremental" clustering. Since we have almost no information regarding to the input distribution which we attained from previous ELMo implementation, it is difficult to decide on the number of nodes to use prior to the conduct of clustering.

However, after reviewing the pseudo-code he suggested in his paper, we concluded it would be too difficult and time-consuming to implement the code by ourselves from the beginning. Fortunately, we discovered a well-designed open source code from the Github written by Adrien Guille. We decided to write the code based on his while changing some parts to make it work more efficiently on our input data.

Most importantly, we decided to improve our code by implementing the GNG-U algorithm instead of simple GNG. GNG-U relocates the "useless" neurons according to its pre-defined utility measure. By adapting the topological structure whenever possible, it can readily track the non-stationary input data distribution. GNG itself has an adaptation parameter, which unfortunately gradually decays leading the network to be "frozen". Regarding such matters, GNG is not sufficient to successfully handle the rapid changes of the input distribution. Thus, regarding the characteristic of our input data which would vary in a high possibility, we implemented a GNG-U algorithm in order to get rid of "dead units" leading the whole network to be unable to handle further inputs.

Also, to improve the clustering algorithm, we decided to use Cosine similarity rather than Euclidean distance measure to detect the similarity among encoded nodes. Cosine

similarity refers to the cosine of the angle between two spots in the input space. It is more representative in the high-dimensional space like ours since it judges the “orientation” not the “magnitude” between two nodes. It is much better to discover how similar two nodes are likely to be in terms of their subject matter. It also allows a lot simpler implementation thanks to its low-complexity.

Following are the results of our GNG algorithm.

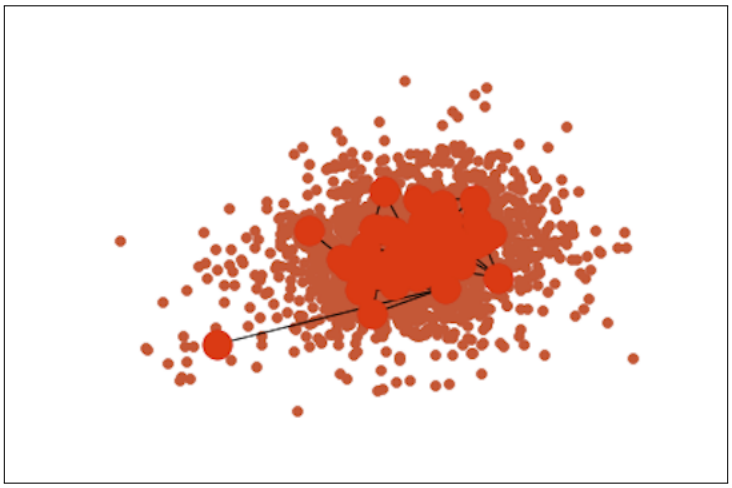

Figure 5: Result of GNG For 2000 Articles

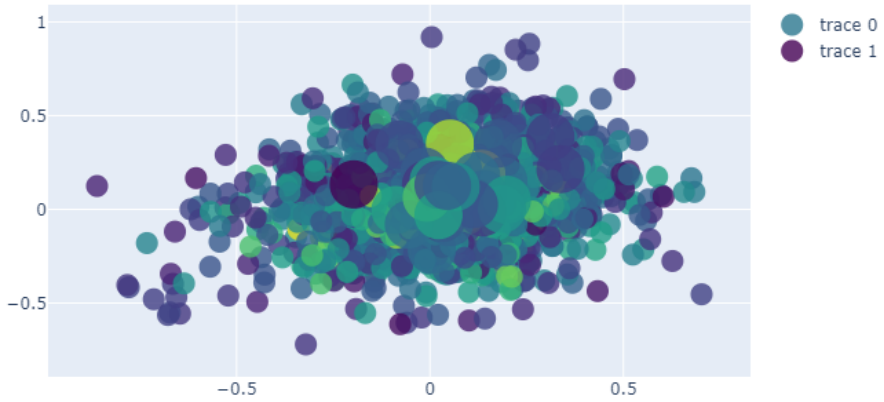

Figure 6: Result of GNG For Labelled 2000 Articles

We realized that there were no meaningful determinants in the clustered result. It was quite different than what we expected with a definite boundaries between clusters. It was due to the high density within the centroid of previously embedded data. We tried to obtain a better clustering by considering skipping normalization process, but realized it resulted from rather more fundamental issues, such as the densely-distributed input data attributes and weakness of our ELMo algorithm.

As the result was not the same as what we predicted, we agreed on that the trend analysis would be impossible to be conducted. Rather, to optimize our result, we decided to continue producing the well-clustered result. The solution we introduced was TextRank algorithm.

TextRank is widely known as the algorithm of which Google search engine is made. The principle is quite simple, but by iterating quite naive conducts a lot, it results in a surprising result. By updating the nodes and edges in a graph, it can readily make a rank of significance within a given input. Every node is assigned weights equally at first. Then, each node shares their assigned weights to its neighbors. For the case below, for instance, A, B, C, D are all assigned 1 for their weight at first, but as they share their weights, each node gets different weight as time goes by.

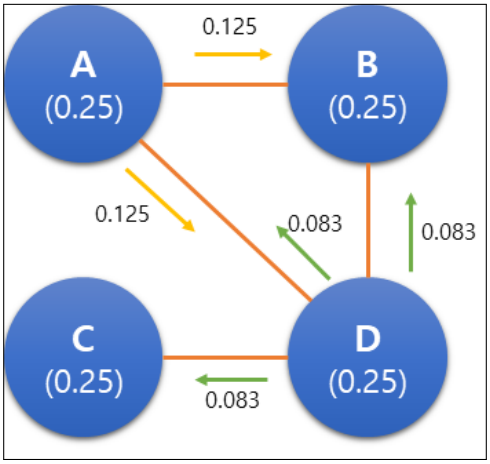

Figure 7: TextRank Algorithm

The updated weight of the vertex is determined by adapting the random surfer model. It adds the sum of the weights taken over from the neighbor nodes and the probability of randomly selected weight of itself. Usually, the node takes over its own weight for 15%. For example, the newly updated weight of B would be $(0.125+0.083)\times 0.85+0.25\times 0.15 = 0.214$. By continuing updating the weights, the weight of each node will converge fairly rapidly.

TextRank, previously called as PageRank, was used for text for the first time in 2004 by Rada Mihalcea and Paul Tarau. We adapted their algorithm for our project. By using a local cluster we have found using GNG algorithm, we considered each neighbor article

as a single node. Also, we assigned the edges among them according to their similarity in the frequency of the words being used.

An example of our TextRank algorithm is shown below. Given 20 articles chosen from the local cluster in order of distance, it shows out 4 representative articles of the cluster.

Most Similar 20 Articles
18 Valentine's Day Gifts For Him That Aren't Ridiculously Cheesy
Tuesday's Morning Email: Inside Alabama's Wild Championship Comeback
21 Creative Valentine's Day Gifts For Her That Aren't Flowers
Wednesday's Morning Email: What You Missed Last Night In Trump's State Of The Union
Articles
[0] Sim: 0.6725904
Headline: Alabama's Day Of Reckoning
Sentence Vector: [0.19114903 0.07474463 ... 0.12134494 -0.24336293 0.04168314]
[1] Sim: 0.64653426
Headline: 33 Anti-Valentine's Day Gifts For People Who Despise Valentine's Day
Sentence Vector: [-0.17134684 0.40277442 ... -0.00901962 0.01219993 0.13390625]
[2] Sim: 0.6423567
Headline: Silent Night, Child-free Night
Sentence Vector: [0.10584313 -0.07349755 ... -0.186909 0.236025 0.07090683]

5 Evaluation

We briefly evaluated the final result which we gained from the TextRank algorithm. It was impossible for us to devise how to evaluate in a more quantitative manner. Rather, we used a more intuitive manner referring to the past statistical data.

First of all, we intuitively checked the temporal continuity among the articles in the same cluster which GNG algorithm had made. If the time slot is rather similar, it would imply that our algorithm somehow manages to cluster the articles in the proper way. Fortunately, the date on which the articles in one of the clusters were written ranged from December 2017 to March 2018. Considering that the assortment of data we have used covered the articles from 2012 to 2018, it suggests that the result of GNG clustering is not quite bad.

Secondly, we checked the contextual relevance between the articles within a cluster. For the same example shown above, the key articles which the TextRank algorithm had sorted were mainly about Valentine Day, Alabama Football Club and Trump Administration. 4 articles were chosen as the key articles, two of which were from the news section

called: Morning Email. This section delivers a daily news as a title of “Monday’s Morning Email”, for example.

Fortunately, we can readily discover the “direct” relevancy between the input articles and the representative ones which TextRank sorted. For example, <Alabama’s Day of Reckoning >itself was chosen as a model article. Also, given 4 articles regarding the Valentine’s Day, <18 Valentine’s Day Gifts For Him That Aren’t Ridiculously Cheesy >was chosen, too.

The most significant strength of our TextRank algorithm was shown by its ability to catch some “indirect” relevancy between articles that showed the similar trend. For example, no less than 8 articles regarding the “holiday” such as Christmas, New Year’s Eve and Mother’s Day were used as the input articles. This also indicates that the clusters GNG formed were very local, but rather powerful. Also, an article about the Philadelphia’s victory in Superbowl, <Philly Streets Run Green After Eagle’s Super Bowl Win >has a similarity with the representative one regarding the football match held in the United States. Furthermore, several articles such as <How Experts and One Congressman Reacted To 2 Days Of Mark Zuckerberg’s Testimony >accords with the articles about Trump Administration in an indirect way in terms of the politics of the United States.

However, we discovered a fatal shortcoming of our project evaluating such results. Given that some articles were from the news section called “Morning Email”, such redundant parts of the input would tangle up the sorting algorithm. Given the fact that the TextRank algorithm sorts out the input by the similarity in the frequency of the word usage, such unnecessary text usage would sneakily give rather malicious impacts on the sorting. We can think of some extreme cases that the representative articles can also be chosen just because it was titled like “Thursday’s Morning Email”. For example, <Thursday’s Morning Email: Inside Matt Lauer’s Downfall >deals with the sexual assault scandal for the former NBC news reader, Matt Lauer. It definitely has no relation with other inputs. Furthermore, considering that two of four articles sorted by the TextRank is from the “Morning Email” section, we concluded that it may have affected the result. So, we attempted to fix this misconducted result by including additional method to exclude such unnecessary part from the TextRank algorithm, but failed to implement due to the shortage of the time. Conclusively, as we were not able to evaluate for other section of our project such as the sentence embedding using ELMo or the GNG algorithm clustering our pre-processed articles, we are not sure which part of our project is mostly responsible for our quite unsatisfactory result.

6 Conclusion

We tried our best to discover an underlying trend by analyzing the newspaper articles. We participated in the weekly meeting with our supervisor and had an additional meeting by ourselves to review what we had done in last week and also decide on what to do for the coming week. In spite of attaining quite a vague result, there were lots of academic gains by managing this project for several weeks.

It was the first time to handle some raw data from the "real-world" to extract implied meanings from a rather unsystematic data. It was a great opportunity to learn and apply some on-site techniques. Based on a number of machine learning methods we had learned from the previous lectures, we sought for an optimal implementation which can best represent our result.

However, there were some weaknesses in our project procedure, and we speculated the reason why our project failed to obtain some meaningful results.

First, in a technical manner, we missed to completely handle the unstructured data. News articles are fully written in text, which would be difficult for us to analyze and interpret its significance solely by themselves. It should have been carefully manipulated in the pre-processing manner. We tried to numerate the articles into computer-readable vectors by using ELMo method. However, conducting the embedding process without any prior knowledge led to a not well-representing result. Rather, we should have thoroughly analyzed the pattern of given data to find the most applicable solution for it prior to the actual implementation.

Second, we had a difficulty in communicating with each other. Due to the pandemic, two of the teammates had left for their home country. We have continued to contact each other using Whatsapp and Zoom meeting. However, there was an apparent limitation using online chat since it was highly vulnerable to be disturbed. Also, the time difference between Sweden and South Korea made it hard for us to work together in the same time slot. It would have been much easier for us if we were allowed to meet face to face to discuss our project.

We ended up without any significant results, however, trend analysis using the text analysis is itself a popular method nowadays. Furthermore, as a number of research has been proceeded regarding the embedding for non-numeral data such as word, video or voice, such unstructured data is in the spotlight as the novel materials that could renovate the conventional trend analysis.

7 References

[1] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations

[2] Bernd Fritzke. Be Busy and Unique ... or Be History – The Utility Criterion for Removing Units in Self-Organizing Networks

[3] Bernd Fritzke. 1995. A growing Neural Gas Network Learns Topologies.

[4] Jim Holmstrom. 2002. Growing Neural Gas: Experiments with GNG, GNG with Utility and Supervised GNG

[5] Trend analysis From Wikipedia

[6] Cosine Similarity From Wikipedia

[7] Michael Phi. 2018. Illustrated Guide to LSTM's and GRU's: A step by step explanation

[8] Prateek Joshi. A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text