Text Mining Term Project Report

Trend Analysis and Issue Tracking

Nguyen Van Giang
20184658
dexter.nguyen7@kaist.ac.kr

Oberwegner Phillip
20186509
p.oberwegner@tum.de

Kalkbrenner Lydia
20186481
lydia.kalkbrenner@campus.tu-berlin.de

ABSTRACT

Natural Language Processing is a domain in computer science which comes under Artificial Intelligence. A number of concerns nowadays related to processing raw text to extract import information. Trend analysis and issue tracking are increasingly becoming promising tasks in NLP for a large amount of data generated day by day especially over the Internet and media. Given a huge corpus of articles, identifying outstanding relations and issues is not such easy work. We first find the top ten issues from a collection of Korean Herald Newspaper Articles from year 2015-2018 where extracted by K-Means Clustering. Three topics were chosen to determine independent and inter-dependent events among articles of all years obtained by measuring cosine similarity using an issue description as query. For each article of the chosen three issues, we extracted information about people, places and organizations involved in the newspaper article by Named Entity Recognition tools that we refined by adding rules. The evaluation section shows the performance of our model and further direction to achieve successes in these domains of NLP.

1 INTRODUCTION

As clustering is a widely used approach as an unsupervised learning approach to extract issues, there are many tools for clustering available. We chose K-Means clustering to tackle the extraction of the top ten issues of Korean Herald newspaper articles using the TFIDF-vector-space model after text preprocessing. To find the top ten most mentioned topics, we assumed that an issue is important, if there are many newspaper articles about that issue, then ranking the issues by Cluster size was chosen as a naive approach, and could be improved by more deep analyses in the future. For the second part of our project, we wanted to extract events within a topic that have follow-up-events. We call the events that don't have follow-up events or are not follow-up events themselves independent events. For the extraction of events, we decided to keep all newspaper articles as events. Doing this we accepted the tradeoff between keeping articles about general discussions or multiple articles about the same events and simplicity. This can also be improved in the future. By analyzing triples obtained by Open Information Extraction, follow-up events were tracked by filtering the triples by keywords that imply follow-up events. The events were obtained by using cosine similarity among the chronologically following articles. We tested the dependency tracking for three issues and decided to describe the results of all three of them instead of only two because they all show different behaviour of our algorithm as we describe in the evaluation.

For tracking the events, we chose three issues to analyze, the reason that we choose three issues instead of two is that we observed and recognized all 3 following issues have the same information significance:

(1) Impeachment of president Park
(2) North Korea
(3) MERS

As we didn't want to rely on the previous clustering result and also wanted to track dependencies over all years, we wrote an issue description and obtained articles that were most similar to the issue description by using Cosine Similarity and the LSI-model to approximate the TFIDF-vector-space model. The gensim library provided a good LSI-model as well as an implementation of cosine similarity.

2 PROBLEM STATEMENT

For **trend analysis**, given a collection of newspaper articles for three years, we should find the top 10 issues for each year and rank them based on prominence and salience, then extracting issues from the article collection. Following tasks are finding a ranking method that can represent the prominence of the issues within the article collection and extracting a description out of the articles that relates to an issue.
In **issue tracking**, from top issues, now picking 2 issues for analyzing, and each issue should identify at least 10 events. For each event, their outstanding attributes need to be extracted. We identify two different types of events: inter-dependent and independent events.

3 RELEVANT WORK

Topic detection and tracking is not a new field, and the domain has mostly been on news sources. Since NIST first proposed the problem of Topic Detection and Tracking (TDT) in the 1990s [2], a lot of work has been done. As the first effort, researchers proposed many approaches to detect and track topics on news documents [4] [3]. Most of these methods are based on either vector space models [9] or statistical models [4] [3]. For example, Yang et al. [9] represented news documents as vectors of words weighted by term-frequency inverse-document-frequency (TF-IDF) and used cosine angle to measure their similarity. Larkey et al. [3] estimated news documents' relevance language model and measured their similarity based on the asymmetric clarity-adjusted divergence. However, there is no common model providing

state-of-the-art results because of ambiguity, untrustworthiness and redundancy of data. This work should explore merits of modern tools in NLP to build a model that mines media news for valuable information the aforementioned domain.

4 DATASET

The given dataset is a collection of Korean Herald Newspaper Articles from years 2015-2018 crawled in JSON format like following: Each article contains 6 sub-fields:

Figure 1: Input dataset in JSON format

- Title: The name of an article
- Author: Author name of an article, could be a person or an organization
- Time: Publishment absolute time to the Internet
- Description: First sentences of the article body
- Body: Content in raw text of an article
- Section: Domain of an article

The dataset can be extended as crawling more data and format into the specified structure.

5 IMPLEMENTATION

An architecture is proposed for Trend Analysis and Issue Tracking. Each problem is solved by combining modules and output of Trend Analysis could be processed to be the input of Issue Tracking task. Some analyses are also conducted to find the most suitable and sufficient solution for each part of solution. The architecture is depicted as following. The model extracts the top trends of 3 years and detects relations among the events as much as possible.

5.1 Trend Analysis

5.1.1 Data Reader. The given corpus is in JSON format, then we use JSON module in Python to read the input data files. As we have several section for an article, dataframe in pandas module is the most sufficient data structure to store dataset. From dataframe, sorting articles by times ascendingly help further steps.

5.1.2 Year-based Categorizer. There are 4 years of article publishments are 2015, 2016, 2017 and 2018. However, 2018's articles are aggregated to 2017 group as the publish date is the first date of 2018. From that point, we counted the numbers of documents for years and separate based on the ascendingly-ordered time article dataframe from data reader.

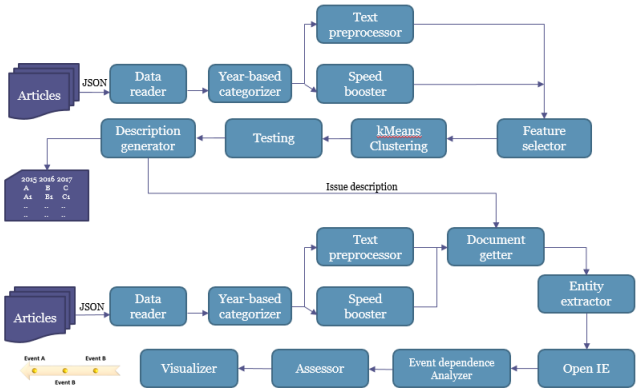

Figure 2: Workflow of Trend Analysis and Issue Tracking

Year	# articles
2015	7156
2016	7485
2017	9128

Figure 3: Numbers of articles per year

5.1.3 Speed Booster. Although a progress bar is constructed along side with processing, the program takes long time in our PC (Intel core i7-6700, 16GB of RAM and 1TB of HDD): about 10 minutes for each year.

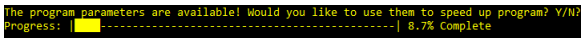

Figure 4: Working of Speed Booster

As we analyzed, most of processing time is dedicated for preprocessing while NTLK has to process token-by-token, so we group unchanged parameters and dumped the data using pickle modules. At least, output of preprocessing and even Named Entity Recognition and Ollie could be done in this way to reduce execution time dramatically. Each time running the program, the program detect existence of pickle files and ask users to use available parameters to boost up the speed. Ignoring this if dataset changed to you want to run the program from the beginning to validate it.

5.1.4 Text Preprocessor. Since we have raw text as input, we need to preprocess data. This stage consists of some tasks:

- Normalization: Convert all text to lower case or upper case to ensure consistency of input. For this task, converting all the words in collection into lower case.

– Tokenization: Chop documents to pieces and throwing away some certain character such as punctuation, white space, using NLTK tokenizer to get tokens from collection.
– Stemming: This technique are popular in preprocessing, but its two sides of the coin, frequently getting meaningless word like “hi”, “ha”, “wa”, then ignoring this technique because trend analysis require fine-grained output from processing.
– Lemmatization: Group together the different inflected forms of a word so they can be analyzed as a single item. The NLTK Lemmatization method is based on WordNet’s built-in morph function.

Each article give a set of tokens, relatively much, then we built vocabulary which is the combination of these token sets, and using stop words to filter insignificant words. With each preprocessing technique is added or removed, the number of entries in vocabulary can be changed slightly with the same input dataset.

5.1.5 Feature Selection. To speed up the clustering, we wanted to reduce the dimensionality of our model by using feature selection. As Sklearn provides an implementation for feature selection by a variance threshold as well as an option to filter by document frequency, we tried out both of these options. However for both implementations we found that it was very difficult to control the amount of features to be selected.

We decided to use document frequency as it was faster and more easy to use.

```
vocabulary length: 15303
vocabulary length after feature selection: 6681
vocabulary removed by feature selection: 8622
```

Figure 5: Vocabulary size by feature selection

5.1.6 Clustering. For clustering the features extracted before, we used the k-means++ implementation of Sklearn. Our resulting parameters for the clustering are 100 iterations, 10 consecutive runs and k equals 149. The 10 consecutive runs define the number of runs the k-means++ runs with different cluster centroid seeds and the best one in terms of the sum of squared distances of each article to their cluster center is the cluster result. However we did not start with k equals 149 at the first place, in order to find this k value we had to run the algorithm several times with different values for k. The k values we tried were between 100 and 200 and we evaluated the best k whilst writing the article headings of our top ten clusters to a file. In addition we wrote the sum of squared distances of the articles to their cluster center to the same file. After, we had a look at the text file and decided if it was a good result or not. However, because we start with random initial cluster centroids, we have a slightly different result with every execution. Further, we decided that our top ten clusters are not the ten clusters with the most articles because with k values between 100 and 200 we end up with some clusters that have more than 400 articles. And for those it was really hard to find a description. Because of that we choose the ten clusters with the most articles but less than 150 articles which gave us a good result and well defined top ten clusters.

5.1.7 Description creation. The descriptions for the top ten issues of each year is done manually by reading the headings of the articles of the top ten clusters and writing a suitable description for them. Our first idea was to use the features of the cluster and use them to automatically write a description, however that was too hard and that is why we ended up doing it manually by hand.

5.2 Issue Tracking

5.2.1 Getting Documents. As we didn’t want to rely on the clustering result of the previous task, we used Cosine Similarity to obtain articles related to an issue.

Gensim provides an implementation of Cosine Similarity as well of LSI (Latent Semantic Indexing), which transforms the documents into a latent space of a lower dimensionality [5]. We wrote a description for each article which is read by our program and use Cosine Similarity to retrieve a user-specified number of documents that are most similar to the description. This list of articles is then used for further analysis.

```
Park Geun-hye impeached
Parliament votes to impeach Park: 234 for, 56 against
Will President Park face impeachment?
Park will not show up at final hearing: lawmaker
Impeachment trial to consider agencies' written statements
Court to face hurdles in impeachment witness questioning
Seoul on high alert ahead of Park ruling
What will Park’s impeachment trial be like?
[From the scene] Rallies sharpen their rhetoric as Park trial nears end
S. Korea protests switch focus to impeachment court
Court hears final arguments in Park's impeachment trial
Saturday's rallies to demand Park and acting president resign
Parliament hands in document supporting Park's impeachment to top court
Court in final preparations to rule on Park impeachment
Supporters of impeached president clash with protesters
Bodyguard questioned at Park impeachment trial
Court hears final arguments in Park’s impeachment trial
Court set to kickstart impeachment deliberation
[Focus] Constitutional Court stacked in favor of Park?
Court to rule on Park on Friday
Park Geun-hye impeachment explained
Park lawyers call court ‘biased’
Park's supporters rally outside court as impeachment ruling nears
Park meets lawyers over impeachment trial
Tension mounts as impeachment ruling looms
```

Figure 6: Result of Similarity Query (Article Headings related to the Impeachment of President Park)

5.2.2 Named Entity Recognition. For each article in corpus, extracting entities such as person name, location, time, organization gives more accurate knowledge about events in the article. Few state-of-the-art tools were taken into consideration, but there are three most famous libraries: Spacy, NLTK and CoreNLP. Below table compares Spacy with CoreNLP and NLTK in NER task. Spacy and CoreNLP are significantly better than NLTK in this task in term of performance.

Package	Precision	Recall	F-Score
spaCy	0.72	0.65	0.69
CoreNLP	0.79	0.73	0.76
NLTK	0.51	0.65	0.58

Figure 7: Performance comparison of NLP tools [1]

However, CoreNLP does not provide Python API, and the performance gap between Spacy and CoreNLP is tolerant. Thus, we sacrificed accuracy to obtain ease and consistency in implementation. The speed of Spacy, although not the most important factor in our work, could also be an advantage of Spacy over CoreNLP. Spacy returns pairs: ***(entity***

Package	Tokenizer	Tagging	Parsing
spaCy	0.2ms	1ms	19ms
CoreNLP	2ms	10ms	49ms
NLTK	4ms	443ms	–

Figure 8: Execution time comparison of NLP tools [1]

, entityType), so a large number of types entity types, but we just captured 5 types: Recognition errors are sometimes

TYPE	DESCRIPTION
PERSON	People, including fictional.
GPE	Countries, cities, states.
DATE	Absolute or relative dates or periods.
ORG	Companies, agencies, institutions, etc.
EVENT	Named hurricanes, battles, wars, sports events, etc

Figure 9: Expected entity types [8]

attributable to Spacy as it detects "Yophap" or "Zika" as person name, "Park" as location, "H5N1" as organization. Unformatted text as input also results in wrong entities. To increase accuracy in NER, rules are added to filter to put entities to their correct places, but this kind of manual work is effort-consuming and domain-specific when we need to look at the result to see the most frequently-committed mistakes and correct them. Spacy still has a long way to go.

5.2.3 Relation Extraction. **OLLIE for Relation Extraction**
Among techniques todays for Information Extraction, we chose Open Information Extraction for its advantages over others. Hand-labeled data (documents on a specificName topic) and pre-specified relations (supervised learning) are required in traditional IE systems. However, such kind of system is limited because of their supervised natured, then not able to scale to word wide web [7]. With the given dataset,

	TRADITIONAL IE	OPEN IE
INPUT	Corpus + hand-labeled data	Corpus
RELATIONS	Specified in advance	Discovered automatically
EXTRACTOR	Relation specific	Relation independent

Figure 10: Comparison between Traditional and Open IE

Open-IE is selected for its merits:

- Avoid hand-labeling data.
- Single pass over corpus.
- No pre-specified vocabulary.

Open Information Extraction (IE) systems extract relational tuples from text, without requiring a pre-specified vocabulary, by identifying relation phrases and associated arguments in arbitrary sentences. However, state-of-the-art Open IE systems, ReVerb (Fader et al., 2011; Etzioni et al., 2011) and WOE^{parse} (Wu and Weld, 2010) share two important weaknesses [6] that degrade the performance of Open IE.

- Expanding the syntactic scope of relation phrases to cover a much larger number of expressions.
- Expanding the Open IE representation to allow additional context information such as attribution and clausal modifiers.

OLLIE extractions obtain a considerably higher yield at higher or comparable precision relative to existing systems while ReVerb seems to be suitable with small dataset, and WOE^{parse} to have lower precision and yield. **OLLIE post-**

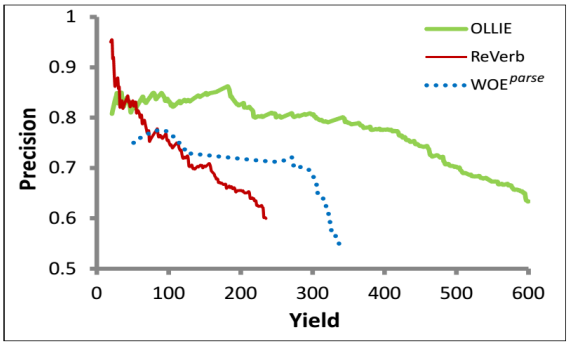

Figure 11: Comparison of different Open IE systems [6]

processing
OLLIE expands and covers much larger of number of relation expressions, thus causing duplicated tuples. Tuples are defined as:

$$t = e_i, r_{i,j}, e_j$$

Where e_i, e_j are entities (strings) and $r_{i,j}$ is the relation between them (string). Furthermore, OLLIE introduces enabling condition which is a condition that needs to be met for the extraction to be true. Certain words demark an enabling condition, such as "if" and "when". Ollie captures enabling conditions if they are present. As we analyzed in the out-

```
sentence: If I slept past noon, I'd be late for work.
extraction: (I; 'd be late for; work)[enabler=If I slept past noon]
```

Figure 12: Structure of OLLIE tuple

put of OLLIE, enabler could help identify relations amongst incident. And as the consequence of expanding expression, OLLIE gives redundancy. To remove redundant tuples for an expression, we used *SequenceMatcher* inspired by Ratcliff/Obershelp algorithm to calculate similarity ratio of two

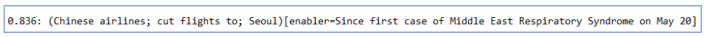

Figure 13: Expanded structure of OLLIE tuple

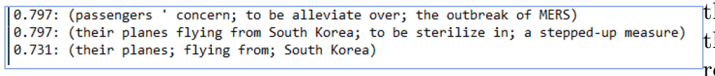

Figure 14: Redundancy in OLLIE output

strings and discarding with keep the highest confidence score tuples given by OLLIE.

$$D_{ro} = \frac{2 \cdot K_m}{|S_1| + |S_2|}$$

Here, $\boldsymbol{K_m}$ is a number of **matching** characters,$|S_1|$ and $|S_2|$ are lengths of strings S_1 and S_2 and respectively, $\boldsymbol{D_{ro}}$ is the distance calculated by the algorithm.
Single pass corpus extractor only generates tuples per sentence, then it is necessary to know which article tuples belong to. Then, we built an *Index Tagger* and *Index Extractor* to get the article index corresponding to tuples. The idea is simple as at the input of OLLIE, a token containing article index (tagger) is attached in article string, and at the output of OLLIE, parsing based on this known token to know article index of tuples (extractor).

5.2.4 Analyzing Dependent Events. As we obtained the triples from OLLIE, we wanted to use the information about entity relations to track dependent events. Therefore we chose only to detect follow-up-events that are mentioned in previous articles in a certain form. We chose to filter the following relations and their derivations from triples to predict a follow-up event:

$$r_{dep} = \{plan, call, urge, allow\}$$

This approach assumes a sentence structure as the following examples that we took from the newspaper collection:

(1) "North Korea urged South Korea to shift its inter-Korean policy."
(2) "The South plans to allow civilian inter-Korean exchanges."
(3) "The prosecutors [...] plan for a face-to-face investigation with Park."
(4) "The Constitutional Court plans to deliver its final ruling on President Park Geun-hye 's impeachment on Friday."

As we filtered the triples that contain the above relations, we formed a search query q concatenating the entities for the triples such that $r_{i,j} \in r_{dep}$ holds:

$$q = e_i e_j$$

The search query is supposed to contain information about the person or institution and an action as for example the cases above:

(1) North Korea South Korea to shift its inter-korean policy
(2) The South civilian inter-Korean exchanges
(3) The court face-to-face investigation with Park
(4) The Constitutional Court deliver its final ruling on President Park Geun-hye 's impeachment on Friday

We used the search query to find the article that is most similar to the search query using Cosine Similarity among the articles chronologically following. The procedure to find the most similar articles was already described above. The result of the search queries above is the following (keeping the same order as above):

(1) S. Korea calls for peace, reconciliation on inter-Korean summit anniversary
(2) Taekwondo event may spark 'sports diplomacy' with NK
(3) Park's lawyers ask Constitutional Court to exclude key evidence
(4) Park's lawyer says to accept impeachment ruling

This approach assumes that the action was planned was actually going to happen and reported in another newspaper article as it is the case for the given examples. If the action didn't happen, the algorithm will only retrieve the article that is most similar to the search query which is not neccessaryly a follow-up event.

5.2.5 Visualization. We used the plotly library to visualize the result. The data points represent events distributed over the time on the x-axis. Dependent events are connected with an edge.

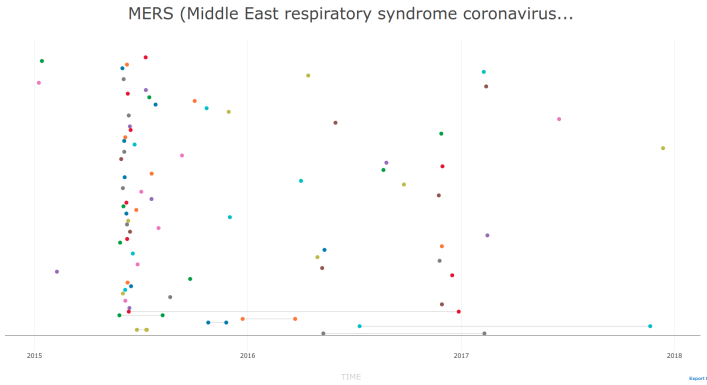

Figure 15: Visualization of MERS Issue Tracking

By hovering over the data points the plot also shows the event data.

6 EVALUATION

6.1 Getting Documents

As we used Cosine Similarity to obtain the related articles for an issue, we measured the precision of this technique.

The precision for the impeachment of president Park is very high, since every obtained article is related to the issue. The model gets confused about MERS because many articles that were retrieved were about bird flu, Zika or other diseases instead of MERS. The two articles that were confused about

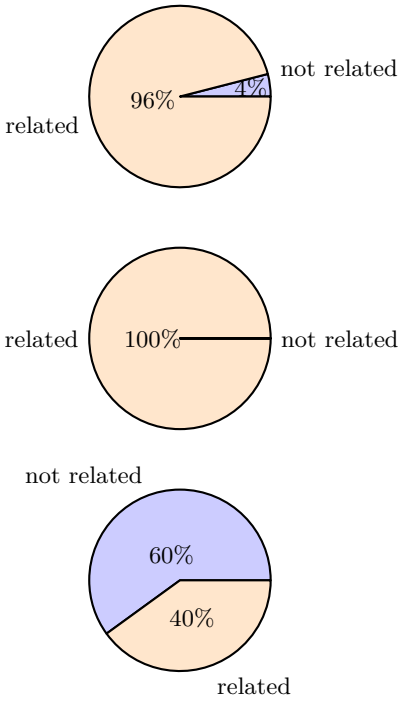

Figure 16: Precision of North Korea (top), President Park (middle), MERS (bottom) issues

North Korea were about Japan, but mention North Korea in the article.

6.2 Clustering

For our needs k-means++ is a good choice because it is fast and efficient in terms of computational cost. Further k-means++ clusters every article to exactly one cluster, while LDA for example clusters each article to several topics. Because we have a look at the article headings to create the descriptions it would be horrible to have an article with a heading about a completely other topic in an cluster, only because it shares some features with the other cluster articles. That would happen if we would have chosen LDA. However, we chose k-means++ and the reason for that is exactly the hard clustering, every article can only be in one cluster. Which compares better to the fact that we only have a look at the article headings to create the issue descriptions. We tried to evaluate our clustering with the Dunn-index, however because we have some clusters with many articles which we are not considering, the resulting Dunn-index would not reflect the quality well enough. In addition we compared the sum of the squared distances of the articles to their closest cluster center for every k with k-1 and k+1. For k to number of articles, this sum gets smaller, but if one has a closer look and picks a small enough interval, one can observe that their exist some local maxima and minima. Such a local minima exists for k equals 149 and that is the reason why we chose 149 over k's close to 149.

6.3 Dependence Tracking

We also measured the precision of the Dependence tracking. The result was quite disappointing because the precision was below 0.5.

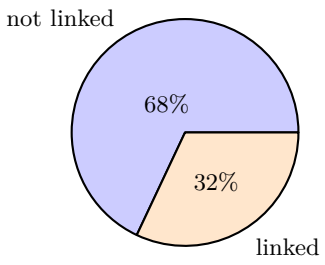

Figure 17: Precision of Dependence Tracking (Impeachment of President Park)

The evaluation was done by reading the article heading and description of the articles to validate if the articles are linked or not. However, this is our only possible way to get as close to the ground truth as we can because we can't read all of the articles to find out if they are really connected.

7 RESULT

The following table holds the descriptions of the top ten issues for every year.

2015
South Korea and the U.S vs North Korea conflict
Criminals
Military tensions and actions
International meetings of South Koreas president
Court judgements
Talks about peace between North and South Korea
MERS outbreak, curing and progression
MERS impacts on Korean politics
Seoul
South Koreas and Japans relation

8 CONCLUSION AND FUTURE WORK

We recognized the popular implementations for stemming are still rule based. Not all the words can be stemmed to their right root word. With stemming, there is lot of ambiguity which may cause several false positives. Besides, with k-means clustering, determining the number of clusters in a data set is time-consuming, and the correct choice of k is often ambiguous. In addition, although giving state-of-the-art result

2016
North Korean missile launches
Sanctions against North Korea
World-wide reactions on North Korea
Parks impeachment and court judgements
Kim Jong-un
THAAD missile defense
Surveys
Political scandal between Choi Soon-sil and Park
Park and nomination of people
Parks resignation and apology

2017 + 2018
International news
Heavy crimes and judgement
Koreas presidential election race
Parks impeachment
Koreas political parties
After the presidential election, the formation of a government
Park and the trial
Korean parliament and ministers
'Comfort women' and sex slavery
Diseases like TB, cancer and mental illnesses

compared with existing tools, OLLIE requires much extra effort for pre-formatting and post-processing. The more structured input are, the more accurate output could be achieved. Name Entity Recognition tools like Spacy easily produce errors because of no deep understanding about entities. In our opinion, applying Nature Language Understanding (NLU) for NER tools would enhance accuracy. When trend analysis could be addressed by combining tools, there is no issue tracking model available because this task is domain-specific and far more complicated regardless of pervasion of data from media. Our approach for the issue tracking is still very simple and has a low precision, but we think that it can be improved by refining the keyword list that indicates a follow up event. Also a similarity threshhold could help to filter redundant results. Ultimately, evaluation is nearly the most important part when constructing a new model, but it is underrated at first. As future work, a comprehensive and appropriate testbed should be built and applied to assess the performance of the new model.

9 CONTRIBUTIONS

In our weekly team meetings we always set weekly goals and tried to distribute the tasks evenly. However, we sometimes had other assignments and delayed our weekly goals for one week. For this reason we didn't fully succeed to distribute the work evenly.

All team-members agreed on the following overall work distribution:

- Giang: 45 % Text Preprocessing, Open Information Extraction, Named Entity Recognition, Code Structure and Documentation
- Lydia: 35 % Feature Selection, Dependency Tracking, Visualization
- Phillip: 20 % Clustering, Description Generation

For the slides and the report we all contributed content about the parts that each of us implemented, while Giang prepared the slides and Lydia prepared the Latex template.

REFERENCES

[1] ANALYTICS VIDHYA. 2018. https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using-spacy-in-python/. Online; accessed 18 Nov 2018. (2018).

[2] David Graff, Chris Cieri, Stephanie Strassel, and Nii Martey. 1999. The tdt-3 text and speech corpus. In *Proceedings of DARPA Broadcast News Workshop*, 57–60.

[3] Leah S Larkey, Fangfang Feng, Margaret Connell, and Victor Lavrenko. 2004. Language-specific models in multilingual topic tracking. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 402–409.

[4] Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas. 2002. Relevance models for topic detection and tracking. In *Proceedings of the second international conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., 115–121.

[5] Radim Řehůřek. 2018. Topics and transformations. https://radimrehurek.com/gensim/tut2.html. Online, accessed 16 Dec 2018. (2018).

[6] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 523–534.

[7] SINA MIRAN. 2018. Brief Introduction and Review of Open Information Extraction (Open-IE) Systems. https://ece.umd.edu/~smiran/OpenIE.pdf. Online; accessed 01 Dec 2018. (2018).

[8] SPACY. 2018. https://spacy.io/usage/linguistic-features. Online; accessed 21 Nov 2018. (2018).

[9] Yiming Yang, Jaime G Carbonell, Ralf D Brown, Thomas Pierce, Brian T Archibald, and Xin Liu. 1999. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems and Their Applications*, 14, 4, 32–43.