CS230

Aamir Rasheed
Department of Computer Science
Stanford University
aamirar@stanford.edu

Dunia Hakim
Department of Computer Science
Stanford University
dunia@stanford.edu

Abstract

To mitigate the issue of misinterpreting events and information due to a lack of contextual information in news stories, our project aims to link new, unseen news articles to related articles in a small dataset of 1224 news stories. To do this, we used a doc2vec model trained on a large wikipedia corpus to generate vector embeddings for our dataset. For our model, we used a clustering approach from where we construct a sparse weighted graph of cosine similarities using the MST-KNN method, then use a Markov stability algorithm to generate the final clusters. While some of our clusters were extremely good, we were limited by the size of our dataset.

1 Introduction

With over-sensationalized news, there is a greater and greater need for verified, fact-based information that the average layperson can digest easily. To mitigate the issue of misinterpreting events and information due to a lack of contextual information, our project sponsor has proposed linking contextually related news articles for any news article. However, there isn't enough manpower to search through data stores of tens of thousands of news articles for the right contextual ones. This is a problem that would be much better solved by algorithms that can relate articles automatically. Our project aims to automatically find relevant contextual articles. More concretely, given a certain article as input, our program aims to output which other articles in the dataset are the most related contextually to the given article.

2 Related work

As will be explained in later sections, for our main model, we decided to cluster the news articles and then classify them in order to get the most similar articles given one article. During our attempts at clustering the articles, we considered related work that worked on hierarchical clustering and works on non-hierarchical clustering. We first considered the following three non-hierarchical clustering methods.

Clustering Urdu News Using Headlines [3] generated similarity scores between each document using a simple word-overlap score. While the algorithm is simple and does not require training, we found it to be too simplistic for our project and decided to use doc2vec embeddings for similarity comparisons instead. Character-level convolutional networks for text classification [4] explored the use of character-level convolutional networks (ConvNets) for text classification. In the paper they compare the performance of character-level convolutional networks against traditional models such as bag of words, n-grams and their TFIDF variants, and deep learning models such as word-based ConvNets and recurrent neural networks. At the end, they found that while character-level convolutional networks could achieve state-of-the-art or competitive results on large- scale datasets, the traditional models did better on smaller datasets. Seeing how our dataset is quite small, we decided

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

to stick to the traditional model bag-of-words. Clustering Short Texts using Wikipedia [5] proposes a method of improving the accuracy of clustering short texts by enriching their representation with additional features from Wikipedia. Empirical results indicate that this enriched representation of text items can substantially improve the clustering accuracy when compared to the conventional bag-of-words representation. While we were interested in the fact that it could produce better clustering than bag-of-words, this method does not generate hierarchical clustering which we eventually decided on.

We then considered the following three hierarchical clustering approaches. Hierarchical Clustering of a Finnish Newspaper Article Collection with Graded Relevance Assessments [2] utilized principal component analysis to reduce the dimensions of the document vectors, then were partitioned using a heuristic rule and generated clusters by means of the average linkage and Ward's method. While the results were good, we ultimately wanted to execute a hierarchical clustering method with a search tree for the specific application we were targeting our project for.

Content-driven, unsupervised clustering of news articles through multiscale graph partitioning [1] performed hierarchical clustering on news articles by first generating a cosine similarity graph between Doc2vec-generated embeddings for each story, pruned the graph by means of the MST-KNN algorithm [8], then applied a multi-scale community detection method (Markov Stability) to partition the similarity graph to generate the clusters. The level of resolution for each cluster can be varied by adjusting the Markov time.

Hierarchical Document Clustering Using Frequent Itemsets [6] proposed to use the notion of frequent itemsets, which comes from association rule mining, for document clustering. The intuition of our clustering criterion is that each cluster is identified by some common words, called frequent itemsets, for the documents in the cluster. Frequent itemsets are also used to produce a hierarchical topic tree for clusters. By focusing on frequent items, the dimensionality of the document set is drastically reduced. This method actually seemed very related to what we hope to accomplish. However, seeing that [1] is also highly related to our work and that it even focuses on news articles too, we decided to follow [1] rather than this paper.

3 Dataset and Features

Our dataset was provided by Jens Erik Gould, who currently is leading a news platform dedicated to unbiased news. There were 1224 articles, and the topics covered were world news and US national news, with little to no reporting in arts and entertainment, sports, or local news. The news articles in the dataset are unique in that they are bare-bones facts of what is being reported, and are therefore generally short. The average word length was 362. A more detailed histogram of the word length is shown below.

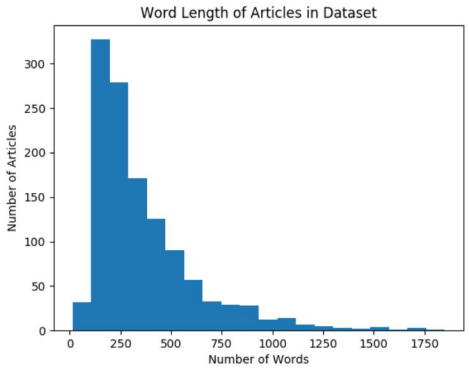
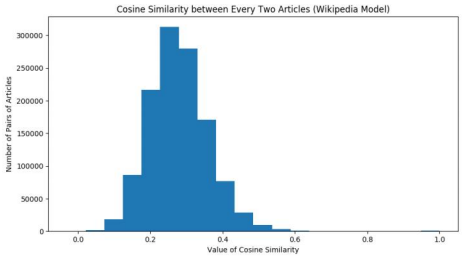


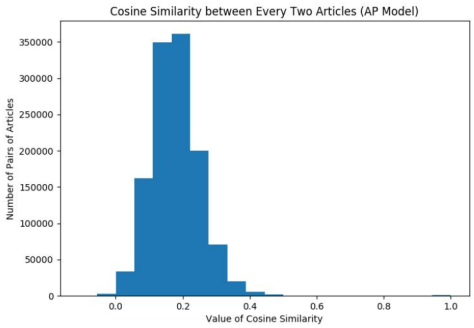
Figure 1: Distribution of Articles' Word Length

For our features, we used a doc2vec model trained on a wikipedia dataset of 35M articles to generate document embeddings for each article in the dataset. However, due to the short length of the articles and little variety in topics covered, many of the vectors were equally related to each other, as pictured in the cosine similarity histogram below. We also had the option of using a doc2vec model trained on

an AP news dataset, but as seen below, the cosine similarity histogram had less variety, and would have ended up making it harder to find good clusters and similarities in the next steps.



(a) AP News Pretrained Model



(b) Wikipedia Pretrained Model

Figure 2: Distribution of Cosine Similarity Between Every Two Articles

4 Methods

For our baseline approach, we used a pretrained Doc2Vec model that was trained with the algorithm distributed bag-of-words on the English Wikipedia corpus. Using this pretrained model and the gensim package, we generated a 300-dimensional feature embedding for each of the articles using the article body. We then calculated the cosine similarity of every pair of stories and generated the top similarities for each story, which allowed us to get the most contextually related articles for each article in the dataset.

Our next step was to implement a more advanced model where we could find related stories given an unseen news article to our algorithm. After discussion with our teaching assistants, we decided on an approach where we would cluster articles, apply that label to the vector embedding, and train a shallow network to learn a mapping between vector embeddings and cluster labels. This shallow network could then be used to infer a cluster label given the vector embedding of a news story generated by our pretrained Doc2vec model.

For our first attempt, we used a simple k-means approach to cluster our points, then when finding the optimal k proved very difficult, used the xMeans [7] clustering method.

After generating several different clusters and varying the parameters with these approaches, we realized that there are many ways to cluster articles. For example, given an article about Poland's supreme court justice nomination and another article about the United States Supreme Court Justice Kavanaugh's nomination, we would not cluster them together if we were looking for all the articles about the Kavanaugh nomination. However, we would cluster them together if we were clustering for supreme court justice nominations around the world. With this in mind, we decided to use a hierarchical clustering approach. We would be able to cluster articles about very specific topics, and have those topics be subtopics in a hierarchy of more and more general clusters. We chose to go with the Markov stability approach from [1] for the reasons mentioned in our Related Work section.

To implement this algorithm, we first generated a weighted cosine similarity graph from our vector embeddings. Then, we generated the MST-KNN graph to develop a more sparse representation. Finally, we used a Markov Stability approach to extract the multi-scale community structure. By varying the Markov time and looking for dips in variation of information, we would be able to accumulate small and specific clusters into large and general ones.

5 Experiments/Results/Discussion

As previously explained, given a certain number n and a given article, our baseline approach outputs the most similar n articles from the dataset to the given article. Thus, in order to quantitatively evaluate the performance of our baseline approach, we randomly chose 50 articles and evaluated each

of the 5 most similar articles that were matched to each of those 50 articles. Through this sample, we found that the percentage of the 50 articles that had "satisfactory" results matched to them was 87%. In other words, we predict that the percentage of "satisfactory" results overall for our baseline approach is 87%. For clarity, we counted the results as "satisfactory" whenever 1. the ranking sequence was reasonable in that related articles were ranked higher than non-related ones (in case there are less than 5 related articles in the dataset), and 2. the related matched articles were among what we would have manually chosen for the given article. Here is an example of a "satisfactory" or a good result:

> Given the article titled "Zimbabweans vote in first election without Mugabe on ballot", the most similar articles in the dataset are titled:
> #1: Zimbabwe's former VP leaves country, citing threats
> #2: Mnangagwa declared winner of Zimbabwe's presidential election
> #3: Emmerson Mnangagwa sworn in as Zimbabwe's president
> #4: Zimbabwe military detains Mugabe, seizes control of government
> #5: Mnangagwa sworn in as Zimbabwe president, after constitutional court approves his election win

According to our quantitative assessment of the baseline approach, we predict that 87% of the results are similar to the example above. Contrastingly, here is a "unsatisfactory" or a bad result:

> Given the article titled "Q&A: A breakdown of the US immigration court system", the most similar articles in the dataset are titled:
> #1: Mueller reportedly subpoenaed Trump financial records from German bank
> #2: AT&T's potential Time Warner acquisition delayed
> #3: AT&T says 'association' with Cohen was 'serious misjudgment
> #4: AG Sessions announces new judicial limits on immigration cases
> #5: UK government announces 3 policy papers on EU exit negotiations

While we are unsure exactly what went wrong in the above unsatisfactory example, our guess is that the word embedding for words with the ampersand character might have wrongly affected with the overall vector embedding for the document and thus made the cosine similarities inaccurate.

As for our clustering attempts, using the kMean and xMean algorithm were both mostly unsuccessful. For instance, xMean only produced one cluster which included all of the dataset. Fortunately, the implementation of [1] which used kNN-MST and Markov Stability gave better clustering results. By plotting the variation of information and the number of clusters at each of the 401 Markov times, we can find which clusters are useful.

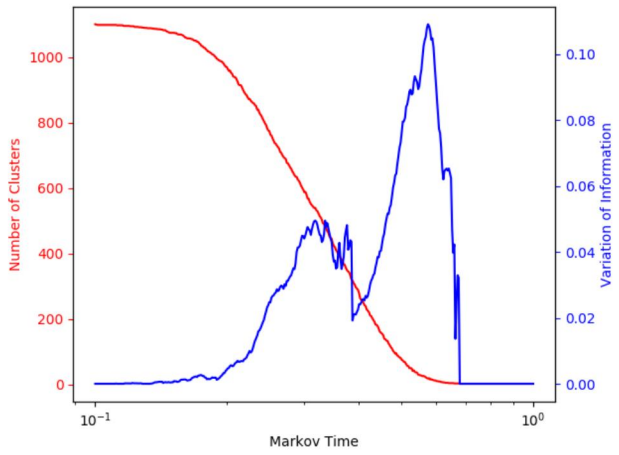


Figure 3: Number of Clusters and Variation of Information As Functions of Markovian Time

Dips in variation of information indicate robust clusters. Thus, according to Figure 3, we had one dip which means that we only had one robust group of clusters. The group of clusters contains 300

clusters, and only 7 of those clusters had more than 7 stories. Of those, only one was poorly clustered, represented below in Figure 4. Additionally, Figure 5 is represents one of the six good clusters.

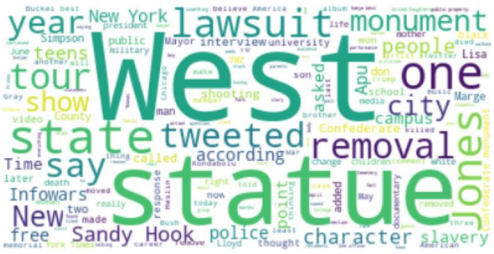


Figure 4: Example of bad cluster. None of the key words in this story are related. Of the seven stories in this cluster, only two were related.

Figure 5: Example of a good cluster. We can see this cluster contains stories pertaining to Trump's trade war with China. 6 of the 7 clusters were like this.

Because we had so few good clusters, and most clusters had just 1 story, we decided to not pursue training a shallow network to classify stories into clusters. Firstly, the clusters would most likely not be trained enough because the output softmax layer had 300 dimensions, and secondly, our dataset wasn't large enough for the network to learn a good representation for a 300 dimensional output.

6 Conclusion/Future Work

Considering the task of finding the most similar articles, in our dataset of 1224 articles, given a certain article, the baseline approach did much better than the current clustering results would allow classification to get. However, since the baseline approach has a time complexity of $O(n)$, it would be very slow and inefficient with larger datasets. Thus, given how well we saw most of the large clusters do, it would be reasonable to assume that for larger datasets, clustering and then classification would be a better and more efficient approach overall.

In order for our final clustering method to work better, we can think of several that we would try if he had more time. First, we would try augmenting our dataset in order to have a dataset size closer to the one used in [1] (around 9,000). Having a larger dataset would lead to bigger clusters which would make them more useful. It would also increase the variance in Figure 2a, and thus allow the kNN-MST algorithm and the Markov Stability algorithm to more easily find and distinguish between the clusters. Additionally, we think that having a small dataset is what caused us to get only one dip in variation of information rather than several in Figure 3. Getting several dips in variation of information is crucial to building a hierarchy of clusters.

A second thing that we would try is using other features in our dataset that we never got to using. For instance, our dataset includes the groups and places discussed in the articles. So we could potentially make use of that info while creating the clusters.

Finally, after getting one or more robust groups of clusters, we would implement a neural network with two or three hidden layers that would classify any new article into the appropriate cluster. This will allow us to hopefully get matches as good as our baseline but to run much faster and more efficiently.

7 Contributions

Both Aamir and Dunia were brainstorming ideas throughout the quarter. Both divided up the work for writing the proposal, milestone, poster, and paper equally. Both also corresponded with our project sponsor.

Aamir implemented the algorithms from [1], specifically the MST-KNN algorithm and the Matlab code to run the Markov stability algorithm.

Dunia found the pretrained doc2vec model, inferred the embeddings for each news story, and calculated the cosine similarities. She ran kMean and xMean algorithm to cluster the articles. She also wrote the code to parse the results of the Markov stability algorithm and generated the graphs and figures we used in our poster and final paper.

Code

Can be found on the following Github repository: https://github.com/aamirrasheed/news-context

Libraries and packages used in our code:

A. Delmotte, S. Y. M. B., M. Schaub (2012). Community detection using the stability of a graph partition. https://github.com/michaelschaub/PartitionStability?fbclid=IwAR2_hkiWk4b1t1C_bTjcsbkRd5Zq3BJbTk4nfzjRoG22EygOzUtoPMsc8Rc.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, *9*(3), 90–95.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed <today>].
URL http://www.scipy.org/

Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Novikov, A. (2018). annoviko/pyclustering: pyclustering 0.8.2 release.
URL https://doi.org/10.5281/zenodo.1491324

Oliphant, T. (2006–). NumPy: A guide to NumPy. USA: Trelgol Publishing.
URL http://www.numpy.org/

Řehůřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (pp. 45–50). Valletta, Malta: ELRA.

References

[1] Altuncu, M. T., Yaliraki, S. N., & Barahona, M. (2018). Content-driven, unsupervised clustering of news articles through multiscale graph partitioning. arXiv preprint arXiv:1808.01175.

[2] Korenius, T., Laurikkala, J., Juhola, M., & Järvelin, K. (2006). Hierarchical clustering of a Finnish newspaper article collection with graded relevance assessments. Information Retrieval, 9(1), 33-53.

[3] Khaliq, S., Iqbal, W., Bukhari, F., & Malik, K. Clustering Urdu News Using Headlines. LANGUAGE & TECHNOLOGY, 89.

[4] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[5] Banerjee, S., Ramanathan, K., & Gupta, A. (2007, July). Clustering short texts using wikipedia. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 787-788). ACM.

[6] Fung, B. C., Wang, K., & Ester, M. (2003, May). Hierarchical document clustering using frequent itemsets. In Proceedings of the 2003 SIAM international conference on data mining (pp. 59-70). Society for Industrial and Applied Mathematics.

[7] Pelleg, D., Moore, A. W. (2000, June). X-means: Extending k-means with efficient estimation of the number of clusters. In Icml (Vol. 1, pp. 727-734)

[8] Patrick Veenstra, Colin Cooper, and Steve Phelps. 2017. Spectral clustering using the kNN-MST similarity graph. In 2016 8th Computer Science and Electronic Engineering Conference, CEEC 2016 - Conference Proceedings. Institute of Electrical and Electronics Engineers Inc., 222–227. https://doi.org/10.1109/CEEC.2016.7835917