Dense vs. Sparse Representations for News Stream Clustering

Todor Staykovski[1] Alberto Barrón-Cedeño[2] Giovanni da San Martino[2] Preslav Nakov[2]

Sofia University, Sofia, Bulgaria
todorstaykovski@gmail.com

Qatar Computing Research Institute, HBKU, Doha, Qatar
{albarron,gmartino,pnakov}@hbku.edu.qa

Abstract.

The abundance of news being generated on a daily basis has made it hard, if not impossible, to monitor all news developments. Thus, there is an increasing need for accurate tools that can organize the news for easier exploration. Typically, this means clustering the news stream, and then connecting the clusters into story lines. Here, we focus on the clustering step, using a local topic graph and a community detection algorithm. Traditionally, news clustering was done using sparse vector representations with TF–IDF weighting, but more recently dense representations have emerged as a popular alternative. Here, we compare these two representations, as well as combinations thereof. The evaluation results on a standard dataset show a sizeable improvement over the state of the art both for the standard F_1 as well as for a BCubed version thereof, which we argue is more suitable for the task.

Keywords: stream clustering · dense representations · sparse representations.

1 Introduction

The rise of Internet has led to the proliferation of news websites. The resulting information overload created demand for systems, such as Google News, that could help organize the news, making them easier to analyze and consume. The process typically starts with event detection; then, events are optionally connected into story lines and arranged temporarily. Event detection in the news stream is an active research direction, which has attracted a lot of attention recently. While a variety of approaches have been proposed, typically it was addressed using clustering, where the articles are represented as vectors in a multi-dimensional space. Traditionally, a sparse bag-of-words (BoW) representation has been used, weighted with term frequency–inverse document frequency (TF–IDF) [LH17,MZCB18], but recently dense representations derived from word embeddings [LM14] have emerged as a viable alternative [BMZ$^+$18,HKMA16].

Below, we compare the use of sparse vs. dense representations, as well as combinations thereof, on a standard benchmark dataset for news stream clustering. As a baseline, we re-implement a pre-existing system: newsLens [LH17]. Then, we improve this system, and we experiment with sparse representations based on BoWs as in [MZCB18]. We further try dense representations based on doc2vec [LM14] as proposed in [BMZ$^+$18]. Finally, we experiment with combining sparse and dense representations. The experimental results on a standard dataset show a sizable improvement over the state of the art both in terms of standard F_1 as well as of a BCubed version thereof, which we argue is more suitable for the task.

In: A. Jorge, R. Campos, A. Jatowt, S. Bhatia (eds.): Proceedings of the Text2StoryIR'19 Workshop, Cologne, Germany, 14-April-2019, published at http://ceur-ws.org

The rest of this paper is organized as follows: Section 2 gives a brief overview of relevant previous work. Section 3 describes the core system based on *newsLens* and the different representations for the news articles. Section 4 presents our experiments and discusses the evaluation results. Finally, Section 5 concludes and points to possible directions for future work.

2 Related Work

The Topic Detection and Tracking (TDT) task aims to help search and organize news-oriented textual material from a variety of broadcast news media. In particular, it asks to cluster news into topics, which are commonly called *stories*. Below, we describe three systems that are closely related to our work here.

Laban and Hearst [LH17] proposed *newsLens*, a system for organizing a large dataset of news articles into a collection of major stories. Their topic detection algorithm creates local clusters based on keywords extracted from the articles. The system further visualizes the story timelines, showing important information about the story. However, the performance of their algorithm was not formally evaluated, and here we fill this gap by testing our reimplementation thereof on a standard benchmark dataset.

Miranda et al. [MZCB18] focused on clustering a stream of news articles in English, Spanish and German. They used vector representation for the articles and built clusters by maintaining two types of centroid functions for each monolingual cluster. For their experiments, they adapted a dataset from [RML$^+$16], and they processed it to turn it into a collection of articles annotated with monolingual and cross-lingual cluster labels, where story clusters are about particular events. They evaluated their model on the test part of their dataset. We use the English partition of the dataset and we compare our results against theirs.

Barrón-Cedeño et al. [BMZ$^+$18] generated short video overviews describing the coverage of the same event by different news outlets automatically. Four modules were involved in the video production: event identification, de-duplication, coverage diversification, and image gathering. The end user was provided with illustrated videos aiming to present different viewpoints for the coverage of the same event. Unlike the above work, their clustering algorithm represents the individual news articles using doc2vec embeddings [LM14].

3 Models

We re-implemented, to the best of our ability, the relevant core modules of the *newsLens* system [LH17], and we consider this reimplementation as our baseline.

The pipeline of *newsLens* has several stages: (*i*) extracting keywords from the articles, (*ii*) identifying local topics (clusters), and (*iii*) merging long-term topics to give rise to stories. Each document is represented as a bag-of-words vector and weighted with TF–IDF. A word w_i is selected as a keyword for document d_j if its TF–IDF score is higher than a manually set threshold T_0^{nl}, where the superscript nl stands for *newsLens*. Two articles are included in the same topic if they share several keywords and have close dates of publication. The articles are grouped into common local topics by building a graph. Articles (a_i, a_j) are linked by an edge in the graph if they have sufficient overlap in keywords: $|kw_i \cap kw_j| \geq T_1$, where T_1^{nl} is another manually-set threshold. The local topics graph is further refined using a community detection algorithm, following the Louvain method [BGLL08]. After the local topics have been created, a topic-matching process is used for stories that might be interrupted in time. Two topics are merged into a story if they do not overlap in time, but are similar in terms of keyword distribution. Each topic is represented by a vector $v(t_i)$ for topic t_i, which contains the number of keywords in all articles about topic t_i. The similarity between a new and an old topic is computed using cosine. If it is above a threshold T_2^{nl}, the topics are merged.

In our algorithm, we consider only the modules for identification of local topics and for matching long-term topics, which we enhance as described below.

3.1 Identification of Local Topics

The local topics clustering module is responsible for creating the preliminary clusters of news articles. Its input consists of overlapping windows, each spanning n consecutive days. The outcome is a graph $G = (V, E)$ where V is the set of vertices —the news articles— and E is the set of edges. An edge between two articles $\{d_i, d_j\} \in V$ exists only if $sim(d_i, d_j) \geq T_1$, where sim is the cosine similarity computed over representations of the articles and T_1 is an empirically set threshold. We explore both sparse and dense vectorial representations.

Table 1. Statistics about the training and the testing partitions of the English part of the news corpus [MZCB18]: shown are the number of articles, the average number of tokens per article, the total number of clusters, and the average cluster size (± standard deviation).

Partition	Docs	Tokens	Clusters	Cluster Size
Train	12,233	434±364	593	21±32
Test	8,726	521±495	222	39±88

For the sparse representation, we compute TF–IDF vectors. The preprocessing in this case consists of case-folding, lemmatization[1], punctuation removal, and stopwording. For the latter, we use a pre-defined list [SDK10] and we combine it with a list of the words with a document frequency higher than 85% in the training set. We consider different parts of the documents to build the vectors: *(i)* the titles, *(ii)* the bodies, *(iii)* and both.

For the dense representation, we compute 300-dimensional *doc2vec* vectors [LM14], which we train on the Signal Media One-million news articles corpus [CAMM16], which contains 265,512 blog articles and 734,488 news articles from 93k unique sources over a period of one month. We use the same preprocessing as before.

We merged the sequence of overlapping local graphs in the order of their creation.

3.2 Community detection

Following [LH17], in order to avoid merging topics that are loosely connected, we further refined the topics graph structure using a community detection algorithm: the Louvain method [BGLL08]. Due to the comparison of the similarities in a spanning window, it is possible for different densely connected topics to connect erroneously. The role of the community detection algorithm is to find the correct assignment of the nodes into communities. It detects communities in networks using a heuristic that maximizes the modularity of the obtained communities. The modularity of a partition is a scalar value between -1 and 1 that measures the density of links inside communities as compared to links between communities [BGLL08].

3.3 Matching Long-Term Topics

The input to this module are the clusters generated across the different n-day windows from the previous module. The objective now is to identify long-term stories that might have been interrupted for a period of time. Two local topics coming from non-overlapping windows are merged if the cosine similarity $sim(t_i, t_j) \geq T_2$, where t_i (t_j) is the mean of all vectors belonging to topic i (j) and T_2 is an empirically set threshold.

4 Experiments and Evaluation

In this section, we describe the dataset, the evaluation measure, the experimental setup, and the evaluation results.

4.1 Dataset

For our experiments, we use the corpus from [MZCB18]. It is a collection of 33,807 news articles with streaming clusters identified across three languages: English, Spanish, and German. We use the English part, which contains 20,959 documents, and the original training–test split. Table 1 reports some statistics about this dataset. The training partition includes articles in the time interval from December 18, 2013 to February 2, 2014, while the articles in the test set come between November 2, 2014 and August 25, 2015. Whereas in the training partition there are numerous gaps with an average of three days without an article, in the test partition the gaps can be as long as 3 months. This is an important factor to take into account when trying to identify long-term topics.

4.2 Evaluation

For evaluating our models, we use the BCubed versions of precision, recall and F_1-measure [AGAV09]. BCubed measures favor solutions that (i) split a cluster that mixes two categories into two pure clusters (cluster homogeneity), (ii) unify two clusters that contain only items from the same category (cluster completeness), (iii) add an item of a different category to an already noisy cluster instead of a pure one, and (iv) make small errors in a big cluster rather than a large number of small errors in small clusters.

[1] We use the NLTK lemmatizer [BLK09].

The BCubed measures are defined as follows. Let L_i be the category of document i and C_i be the cluster of document i. The relation correctness between documents i and j is calculated as follows:

$$Correctness(i,j) = \begin{cases} 1, \text{ if } L_i = L_j \text{ and } C_i = C_j \\ 0, \text{ Otherwise} \end{cases} \tag{1}$$

The BCubed precision of a document is the proportion of documents in its cluster that belong to the same category, including itself. The BCubed recall of a document is the proportion of documents belonging to the same category that appear in its cluster. The overall BCubed precision and recall are computed as follows:

$$\begin{aligned} \text{BCubed } P &= Avg_i[Avg_{i.C_i=C_j}[Correctness(i,j)]] \\ \text{BCubed } R &= Avg_i[Avg_{i.L_i=L_j}[Correctness(i,j)]] \end{aligned} \tag{2}$$

Additionally to these measures, we also report the standard versions of precision and recall:[2]

$$P = \frac{tp}{tp+fp} \qquad R = \frac{tp}{tp+fn} \tag{3}$$

where tp, fp, and fn stand for true positives, false positives, and false negatives, respectively.

Both the BCubed and the standard F_1-measure are computed as the harmonic mean of precision and recall:

$$F_1 = 2 * \frac{P * R}{P + R} \qquad \text{BCubed } F_1 = 2 * \frac{\text{BCubed } P * \text{BCubed } R}{\text{BCubed } P + \text{BCubed } R} \tag{4}$$

4.3 Experimental Setup

We tuned the hyper-parameters of our model using a temporal 4-fold cross-validation on the training set. We used grid search optimizing for F_1, exploring $n = [4, 5, 6]$ for the size of the sliding window, and $T_1 \in \{0.10, 0.11, \ldots, 0.50\}$ and $T_2 \in \{0.60, 0.61, \ldots, 0.90\}$ for the thresholds. We found different values for T_1 for the different experiments, due to the different models for identification of local topics; yet, we always obtained the same values for $n = 6$, $T_2 = 0.8$ and 50% window overlap, as best performing in the preliminary tests.

4.4 Results

Table 2 reports the results obtained on the test partition. For each model, only the results with the best tuning during the cross-validation are shown. We include two baselines. The first one is our implementation of *newsLens*, using the same parameters as in [LH17]: $T_1^{nl} = 3$ and $T_2^{nl} = 0.8$. The second baseline is the out-of-the-box system of [MZCB18], which is shared in their repository. In all cases, except for [MZCB18], we use Louvain's community detection algorithm (cf. Section 3.2).

We explored applying the TF–IDF sparse representation to the title, to the contents, and both to the title and to the contents of the articles. We ontained best results for the latter, and these are the ones we report in the TF–IDF rows (with $T_1 = 0.31$). This representation yielded BCubed $F_1 = 94.41$, outperforming all other models.

We conducted several experiments using the dense doc2vec representation, including an embedding model trained on the training dataset only. For this, we needed a larger reference corpus [CAMM16] to achieve competitive results, but still not as strong, with a BCubed $F_1 = 89.96$ (with $T_1 = 0.32$). Whereas the TF-IDF model yields roughly twice as many clusters that actually exist in the dataset, this model generates 785 clusters. This is a clear sign that the model fails to merge sub-clusters into single stories.

Finally, we combined the sparse and the dense vector representations in two ways: (i) concatenating the two vectors, and (ii) combining the output of the TF–IDF and of the doc2vec-based clusters. The latter worked better, and this is what we report in the table. The best performance is obtained with $T_1 = 0.32$ for the TF–IDF representation and $T_1 = 0.25$ for the doc2vec representation: BCubed $F_1 = 92.97$.

As the grid search to tune the best thresholds can be computationally expensive, we also tried converting the task into a supervised problem. We represented pairs of documents on the basis of their similarities computed on the TF–IDF and on the doc2vec representations and we labeled them as true if they originally belonged to the same cluster, and as false otherwise. We further divided this resulting dataset into training and testing partitions to train a logistic regression (LR) model with a sigmoid activation function.

[2] We use the implementation released with the corpus [MZCB18].

Table 2. Results for the different models on the test partition in terms of standard and BCubed F_1-measure, precision, and recall. The last column reports the obtained number of clusters generated by each model; the actual number of clusters in the dataset is 222.

	BCubed						
model	F_1	P	R	F_1	P	R	clusters
baselines							
newsLens [LH17]	89.76	94.37	85.58	95.09	95.90	94.30	873
Miranda et al. [MZCB18]	92.36	94.57	90.25	94.03	98.14	90.25	326
unsupervised							
TF-IDF	**94.41**	95.16	93.66	**98.11**	97.60	98.63	484
doc2vec	89.96	93.00	87.12	95.44	95.55	95.34	785
TF-IDF & doc2vec	92.97	95.75	90.34	97.61	97.73	97.48	663
supervised (LR)							
TF-IDF	94.30	94.87	93.73	98.08	97.46	98.71	485
TF-IDF & doc2vec	93.67	92.71	94.65	97.15	95.39	98.98	431

The last two rows of Table 2 show the results of the experiments using logistic regression with TF–IDF similarity alone and in combination with doc2vec similarity as input. The supervised model TF–IDF & doc2vec shows an improvement of over 0.7 points absolute in terms of BCubed F_1-measure over the standard approach (93.67 vs. 92.97). This model is the one that generates the least clusters, a number that is also closer to the actual number to be found. On the other hand, the experiment with supervised TF–IDF achieved worse BCubed $F_1 = 94.30$, comparing to the unsupervised BCubed $F_1 = 94.41$.

Finally, in order to investigate the impact of the Louvain community detection algorithm, Table 3 shows the results of applying it on the TF–IDF model in two ways: (*i*) before and (*ii*) after merging two local topics graphs (cf. end of Section 3.1). We can see that the latter works better; we observed similar results for most of our models.

Table 3. Result of applying Louvain's community detection algorithm on the TF-IDF clustering model before vs. after the local topics merging.

	BCubed						
	F_1	P	R	F_1	P	R	clusters
before	91.55	93.76	89.44	94.36	95.55	93.20	488
after	94.41	95.16	93.66	98.11	97.60	98.63	484

5 Conclusion and Future Work

We have presented an algorithm for clustering news streams, which is an improved version of a previously proposed general framework. We further compared sparse vector representations with TF–IDF weighting vs. doc2vec-based dense representations, as well as combinations thereof. The evaluation results on a standard dataset have shown sizable improvements over the state of the art both for the standard F_1 score as well as for a BCubed version thereof, which we argue is more suitable for the task.

In future work, we plan to improve the process of topic matching. This would require us to build a dataset with news stories that seemingly die, but then reemerge with large gaps in time.

References

[AGAV09] Enrique Amig, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12:461–486, 2009.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[BLK09] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python.* OReilly Media Inc., 2009.

[BMZ$^+$18] Alberto Barrón-Cedeño, Giovanni Da San Martino, Yifan Zhang, Ahmed M. Ali, and Fahim Dalvi. Qlusty: Quick and dirty generation of event videos from written media coverage. In *Proceedings of the Second International Workshop on Recent Trends in News Information Retrieval*, pages 27–32, Grenoble, France, 2018.

[CAMM16] David Corney, Dyaa Albakour, Miguel Martinez, and Samir Moussa. What do a million news articles look like? In *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval*, pages 42–47, Padua, Italy, 2016.

[HKMA16] Kazuma Hashimoto, Georgios Kontonatsios, Makoto Miwa, and Sophia Ananiadou. Topic detection using paragraph vectors to support active learning in systematic reviews. *Journal of Biomedical Informatics*, 62, 06 2016.

[LH17] Philippe Laban and Marti Hearst. newsLens: Building and visualizing long-ranging news stories. In *Proceedings of the Events and Stories in the News Workshop*, pages 1–9, Vancouver, Canada, 2017.

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *ICML '14*, pages 1188–1196, Bejing, China, 2014.

[MZCB18] Sebastião Miranda, Arturs Znotins, Shay B. Cohen, and Guntis Barzdins. Multilingual clustering of streaming news. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 4535–4544, Brussels, Belgium, 2018.

[RML$^+$16] Jan Rupnik, Andrej Muhic, Gregor Leban, Primoz Skraba, Blaz Fortuna, and Marko Grobelnik. News across languages - cross-lingual document similarity and event tracking. *J. Artif. Intell. Res.*, 55:283–316, 2016.

[SDK10] Benjamin Stone, Simon Dennis, and Peter J. Kwantes. Comparing methods for single paragraph similarity analysis. *Topics in Cognitive Science*, 3:92–122, 2010.