AUTOMATIC STORY CONSTRUCTION FROM NEWS ARTICLES IN AN ONLINE FASHION

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Özgür CAN

July 2019
İZMİR

We approve the thesis of **Özgür CAN**

Examining Committee Members:

Asst. Prof. Dr. Selma TEKİR
Department of Computer Engineering, İzmir Institute of Technology

---

Asst. Prof. Dr. Özgü CAN
Department of Computer Engineering, Ege University

Asst. Prof. Dr. Serap ŞAHİN
Department of Computer Engineering, İzmir Institute of Technology

16 July 2019

Asst. Prof. Dr. Selma TEKİR
Supervisor, Department of Computer Engineering
İzmir Institute of Technology

Assoc. Prof. Dr. Tolga AYAV
Head of the Department of
Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude and deep appreciation to my supervisor Selma TEKİR for her constant encouragement. I am grateful for her support and patience throughout the study period.

A special thanks to the members of my thesis defense jury: Serap ŞAHİN and Özgü CAN for accepting to review my work.

I would like to express my deepest gratitude to my parents and sister for their unconditional support.

ABSTRACT

AUTOMATIC STORY CONSTRUCTION FROM NEWS ARTICLES IN AN ONLINE FASHION

Every day, thousands of local and global news become online. Each arriving news piece can have a connection with some previous information, but in a large-scale news flow, it is quite difficult for readers to integrate news and evaluate the agenda in the light of past. Thus, grouping news in a coherent way to construct news stories is a fundamental requirement.

To meet this requirement, first of all meaningful representation of documents on which the clustering is performed must be extracted, and the new story clusters must be generated on the fly in an online fashion.

In this work, we analyze the complex relations of the news articles, and propose a system to generate continuously updated news stories in online fashion. As part of the experimental validation, we provide a step by step construction of a meaningful news story out of news articles that are coming from different sources. The constructed news stories demonstrate the usefulness of the developed system.

ÖZET

HABER MAKALELERİNDEN ÇEVRİMİÇİ ŞEKİLDE OTOMATİK HİKAYE OLUŞTURULMASI

Her gün, internet üzerindeki haber kaynaklarından binlerce yerel ve küresel haber yayınlanmaktadır. Gelen her yeni bilgi, önceki bilgilerle bir bütünlük oluşturur, ancak okuyucuların, büyük ölçekteki haber akışı içinde, bu bütünlüğü yakalamaları ve gündemi, geçmiş ışığında değerlendirebilmeleri oldukça zordur. Bu nedenle haberleri tutarlı bir şekilde birleştirerek, bir haber hikayesi halinde sunmak oldukça önemli bir gereksinimdir.

Bu gereksinimi karşılamak için öncelikle haber makaleleri, anlamlı bir şekilde temsil edilmeli ve yeni hikaye kümeleri anında çevrim içi olarak oluşturulmalıdır.

Çalışmamız kapsamında, bir haber hikayesi oluşturan haberlerin karmaşık ilişkileri incelenerek, birçok kaynaktan toplanan haberleri, bir haber akışı üzerinden otomatik olarak haber hikayeleri halinde kümelendirmeyi sağlayan bir sistem geliştirilmiştir. Deneysel doğrulama kapsamında, belirlenen bir senaryo dahilinde farklı haber kaynaklarından gelen haber dokümanları üzerinde değerlendirme yapılmış ve sistemin başarımı gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

LIST OF ABBREVIATIONS

TDT . . . . . Topic Detection and Tracking
TF-IDF . . . . . Term Frequency-Inverse Document Frequency
TF . . . . . Term Frequency
IDF . . . . . Inverse Document Frequency
CBOW . . . . . Continuous Bag of Words
DBOW . . . . . Distributed Bag Of Words
PV-DM . . . . . Distributed Memory Model of Paragraph Vectors

CHAPTER 1

INTRODUCTION

The digital transformation of news media increased the speed of news delivery and ease accessibility of the news. Nowadays, there are many online news sources that are constantly broadcasting what is happening around the world. With the increasing amounts of data that readers come across throughout one day, it becomes harder for readers to sift through the relevant information quickly. Thus, it is essential to group similar articles together so that readers can have a more comprehensive view and evaluate the agenda in light of the past.

Event detection problem is initiated by the *Topic Detection and Tracking (TDT)* project (Allan (2002)) and widely addressed in many research topics and TDT applications. In the original paper of TDT, a *story* was defined as a single document that may include more than one *event* that happened at a particular place and time, and a *topic* was considered as an important event together with all related events. After the TDT's original paper, terminology and understanding of particular notions evolved. Many researches on news chain construction and clustering have been using different terminologies. One common terminology that we also follow in this work, defines *topics* as local groups of news articles, that can be grouped into a *story*, in time.

A news story usually consists of many news articles discussing multiple related events. Although a story can start and end in a relatively short period in time, such as *2016 Nice truck attract*, some stories can last for years and evolve in time, such as *Facebook-Cambridge Analytica data scandal* which was first reported by the journalist Harry Davies in December 2015, later on lots of people involved the case until it became a major political scandal in early 2018. The task of story detection is a process of grouping related news articles in a coherent way. Many researches address to this problem but the issue of effectively detecting and tracing news stories from large volumes of news articles in online fashion remains challenging.

A set of news articles in a small time interval (e.g. 4 days) have potential to create a densely-connected local group around a *topic* which, can then be merged in a *story*. In this work, we present a method for aggregating news articles into topics and then merge them into story clusters in online fashion from a continuous stream.

There are three main tasks in extracting the topics: getting a meaningful numerical representation of a document, finding out the similarities between document representations, and clustering the similar documents.

To represent news articles, we use doc2vec(Le and Mikolov (2014)) to have a contextual model. We train the doc2vec network with all the Wikipedia articles, and using the trained model, we extract the representation for a given news article. For the whole set of articles, we create a fully-connected weighted graph, where nodes are vector representations of documents and weights are cosine similarities between them. Having a fully-connected document network gives an opportunity to consider all the document relations without discarding any of them, though it boosts the cost of processing. As weights are the similarities between documents, similar documents are connected more strongly than the others, and strongly-connected news articles have potential to represent a topic. This behavior in document networks can be explained by community structures in networks. In network theory, a community is defined as a partition of a network whose nodes are more densely connected with each other than with any node belonging to a different partition (Girvan and Newman (2002)). Many real-world networks such as social networks and biochemical networks have been shown to possess community structures. Inspiring from this, we propose to find analogous communities in document networks and intuitively we expect communities to represent stories. Many methods have been proposed to detect community structures. With the requirement of finding communities in a fully-connected weighted graph, we use Louvain community detection algorithm (Blondel et al. (2008)) to find out the stories. Document embedding allows us to perform vector operations over the articles. Thus, a story could be represented by adding up all document vectors belonging to the story. To process a large amount of data in an online fashion, we propose a novel sliding window approach named *Inching Window* that allows us to process current news in the context of the recent data. Then, we extend our proposed clustering algorithm to process the news articles in real-time. We use an inching window setup to detect the recent topics. Community detection algorithm is applied to merge the detected topics with existing stories or create new stories. As part of the experimental validation, we provide a step by step construction of a meaningful news story and provide a scenario based experiment to demonstrate the usefulness of the developed system.

The main contributions of this work can be summarized as follows:

(a) novel way to represent stories as vectors.

(b) community-based story detection algorithm

(c) novel sliding window approach that reduces the time complexity of processing each item in the context of the recent data.

(d) online clustering method, which can automatically detect new stories from a continuous stream.

1.1. Organization of Thesis

The organization of this thesis is as follows:

1. Chapter 2 gives definitions for some concepts and algorithms to gain the basic background needed to understand the proposed system.
2. Chapter 3 highlights some online news article clustering methods in literature.
3. Chapter 4 introduces our methodology for representing news articles as vectors and clustering the news articles into stories in an online fashion from a continuous stream. Furthermore, a clustering process on a sample news article corpus is explained through a visualization.
4. Chapter 5 shows the test results and the performance evaluations.
5. Chapter 6 gives a summary of the thesis and suggestions for future researches.

CHAPTER 2

BACKGROUND

2.1. TF-IDF

TF-IDF, which stands for ”term frequency - inverse document frequency”, is a statistical measure that is used to evaluate importance of a term based on its appearance count in the document, and a given document set.

The intuition behind the TF-IDF is combining ”term frequency” and ”inverse document frequency” concepts to generate an importance weight for each word in each document.

Term frequency (TF) measures the appearance frequency of a term in a document to indicate the significance of the term within the overall document.

$$tf(d,t) = \frac{\textit{Number of times term t appears in a document}}{\textit{Total number of terms in the document d}} \tag{2.1}$$

Inverse document frequency (IDF) is proposed by Jones (1988) as a measure of a word's importance. IDF weighs down the importance of the frequent terms while scaling up the terms that are not used very much in a collection of documents.

$$idf(t) = \ln(\frac{\textit{Total number of documents}}{\textit{Number of documents with term t in it}}) \tag{2.2}$$

The TF-IDF assigns a weight to term t in document d given by Equation 2.3

$$TF - IDF(t,d) = tf(t,d) \cdot idf(t) \tag{2.3}$$

Table 2.2 shows the TF-IDF calculation for the given two sentences: S_1: *Turkey will boost security* and S_2: *Brexit will boost tourism*. TF-IDF highlights *Turkey* and *security* words as signature words in the first document, *Brexit* and *tourism* in the second. As *will* and *boost* terms are frequent all over the document set, the idf weighs them down.

term	S_1	S_2
Turkey	1	0
will	1	1
boost	1	1
security	1	0
Brexit	0	1
tourism	0	1

Table 2.1. Term occurrences of the sentences: S_1: *Turkey will boost security* and S_2, *Brexit will boost tourism*

term	tf		idf	TF-IDF	
	sentence			sentence	
	S_1	S_2		S_1	S_2
Turkey	1/4	0	$\log(2/1) = 0.3$	0.075	0
will	1/4	1/4	$\log(2/2) = 0$	0	0
boost	1/4	1/4	$\log(2/2) = 0$	0	0
security	1/4	0	$\log(2/1) = 0.3$	0.075	0
Brexit	0	1/4	$\log(2/1) = 0.3$	0	0.075
tourism	0	1/4	$\log(2/1) = 0.3$	0	0.075

Table 2.2. TD-IDF values of the sentences given in Table 2.1

2.2. Word2vec

Word2vec is a method proposed by Mikolov et al. (2013) to efficiently create word embeddings. Word2vec is a neural network that contains two layers. The input of the network is a sequence of sentences and output is a set of word feature vectors.

The intuition behind the word2vec is to detect similarities between word vectors mathematically and group them together in a vector space. Thus, word2vec generates distributed representations of word features, named *"neural word embeddings"*.

The selection of different model parameters and different corpus sizes affects the quality of a word2vec model. Accuracy of a word2vec model can be improved by optimal selection of the model architecture, optimizing the training data set, the number of vector dimensions and the window size of words.

Word2vec constructs word embeddings, either using context to predict a target word ("Continuous Bag of Words" model) or using a target word to predict the context words. ("Skip-Gram" model)

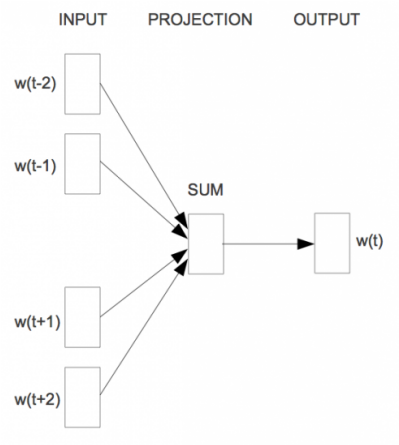

Figure 2.1. The CBOW network model

In the CBOW, context is represented by surrounding words for a specified target word, for instance, given a training sentence *"Ronaldo scores free-kick from an absurd angle"*, *"Ronaldo"* and *"free-kick"* could be used as context words of *"scores"*. Figure 2.1 shows the illustration of the CBOW network model.

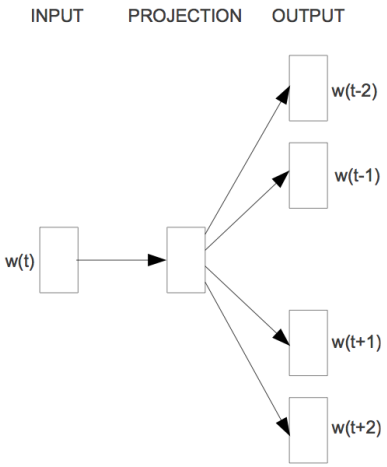

Figure 2.2. The skip-gram network model

Skip-gram model learns to guess the context words from a given target word. Output of the skip-gram is the probabilities of the words appearing around the target word, for instance, for the sentence *"Ronaldo scores free-kick from an absurd angle"*, from the target word "scores", the network tries to predict *"Ronaldo"*, *"free-kick"*, *"from"*, *"an"*, *"absurd"* and *"angle"* words. The network will learn statistics from the count of

occurrences of each word-pairing. The skip-gram network model is illustrated in Figure 2.2.

2.3. Doc2vec

Doc2vec is proposed by Le and Mikolov (2014) as an extension to Word2vec to learn document embeddings by extending learnings from word embeddings. Doc2vec can construct the document embeddings based on two models which are ”Distributed Bag of Words” (DBOW) and ”Distributed Memory Model of Paragraph Vectors” (PV-DM)

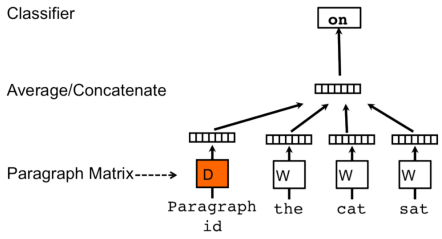

Figure 2.3. The PV-DM network model

The PV-DM model works in the same way as CBOW model in word2vec, but with a small extension. A unique document identifier is added as another feature vector to the input later. Thus, in the end of the network training, the result holds a numeric representation of the document, besides the word representations. PV-DM model is illustrated in Figure 2.3.

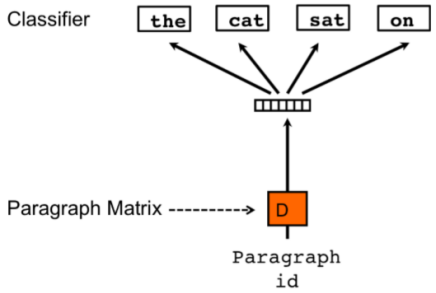

Figure 2.4. The DBOW network model

The DBOW model is an extension to word2vec skip-gram model. The DBOW

model learns to guess probabilities of words randomly sampled from the paragraph in the output. The network model is illustrated in Figure 2.4.

2.4. Louvain Community Detection Method

Communities in networks are groups of nodes that are connected each other more densely than outside the group. Community detection refers to the procedure of partitioning a network into communities of densely connected nodes (Girvan and Newman (2002)). One popular way to evaluate the quality of the partitions is measuring modularity. Modularity is a metric that was proposed by (Girvan and Newman (2002)). It quantifies the strength of division in a network by comparing the density of links inside communities with links between communities.

The Louvain method for community detection is an algorithm proposed by Blondel et al. (2008) to extract communities in networks by maximizing the modularity score for each community.

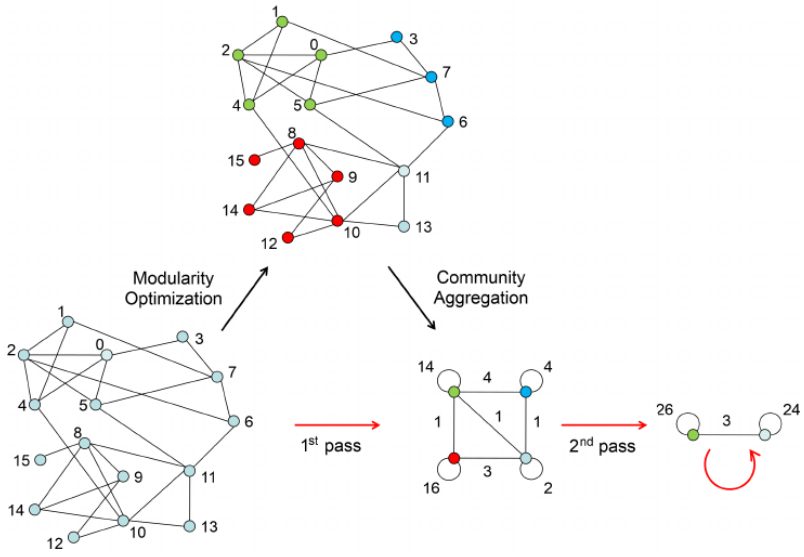

Figure 2.5. Visualization of Louvain algorithm's steps

The Louvain method consists of two steps that are applied repeatedly;

(a) a *greedy* assignment of nodes to communities, and performing modularity optimizations by allowing local changes of communities.

(b) aggregation of the found communities in order to build a community network.

These two steps are repeated until no further increase in modularity. The steps of the Louvain algorithm is shown in Figure 2.5.

The Louvain algorithm is one of the fastest modularity-based community detection algorithms (Blondel et al. (2008)). It also works well with a large number of data points (Blondel et al. (2008)). The method also produces the community hierarchies that we can zoom-in to discover sub-communities.

CHAPTER 3

RELATED WORK

Topic Detection and Tracking (TDT) is an essential research area that is initiated by Allan (2002), and followed by numerous research. TDT research primarily focuses on identifying and following events from a constantly arriving stream of text from multiple sources. In the original TDT work, a story is defined as an individual news article, an event is defined as something that happens at a particular time and place, and a topic is defined as a theme that is related with a set of events. After the original work, terminology and understanding of some notations have evolved, but still, there is no accepted unified terminology in use. The overall purpose of the TDT is breaking the text into individual stories, spotting events that haven't been seen before, and grouping related news articles that discuss the same event.

The problem of topic detection and tracking has been extensively studied, especially with information retrieval techniques. In their study, Can et al. (2010) address new event detection and topic tracking problems in a large-scale multi-source Turkish TDT test collection that is constructed by human annotators. For the new event detection task, a sliding time-window concept is used. For each document, the highest TF-IDF weighted terms are selected for the vector representation. Then, the first story of a new event is defined if the maximum cosine similarity between the newest story and the time-window stories is less than a predefined threshold. For the task of topic tracking, a similar procedure is followed. The cosine similarity between the current story and a target topic which is defined as the centroid of the documents belonging to the topic results in a confidence score. If the confidence score is above a predefined threshold, then the document is classified as an on-topic story and vice versa. For adaptive topic tracking, the centroid term of a topic is updated after assigning a document to the topic. Authors also provide benchmark observations for the new event detection and topic tracking methods applied in a Turkish corpus. Results show that while topic tracking in Turkish dataset has a performance similar to those in other languages, new event detection is less effective than topic tracking in Turkish, and still have room for improvement.

Most popular way in topic tracking is first extracting text features, traditionally with the bag of words and TF-IDF models, to represent the news articles and then grouping related news articles according to their similarities and publication dates by applying

a clustering algorithm. However, in the past few years, different approaches have been proposed to improve detecting and presenting news story chains. Goyal et al. (2013) focus on easing for the readers to explore different contexts of a news story. All the entities appearing in a given document set are extracted by OpenCalais API (OpenCalais (2019)), and sanitized by using an ontology based approach (YAGO (2019)). The news articles are represented, based on Choudhary et al. (2008) study, as a directed graph which is constructed by mining all transformations (Birth, Continue, Merge, Split and Cease), then finding a max-weighted edges covering these transformations. With an extension of augmenting new arriving articles on to an existing graph, the authors propose to construct the graph from a streaming input of articles in adaptive time windows. This graph structure allows spotting all news chains that represent the unique context of a news story. Markov graph clustering is applied to the graph to detect the news clusters which are considered as events, and dense nodes clusters are interpreted as important events. Furthermore, the authors provides a web interface for news events visualization by a timeline panel.

There are also few approaches which took advantage of community detection techniques. In their study, Shahaf et al. (2015) use document keywords co-occurrences to contract a keyword graph and apply a community detection algorithm to detect news stories on the word co-occurrence graph. Inspiring from the structure of the cartographic maps, the authors used ”metro map” metaphor to describe relations between different pieces of information, and ”metro stop” metaphor to describe a cluster of articles. The study defines two metrics for story evolution which are coherence and coverage and tries to maximize the topic diversity while keeping the storylines coherent. The proposed algorithm starts with finding an article set from a query. Then, the articles are segmented into time windows and for each window, article relations are represented by a word co-occurrence graph, a community-detection algorithm is applied to find “metro stops”. After detecting communities, coherent lines are computed. As enumerating all the candidate lines is not computationally feasible, the authors proposed to create coherent lines by constructing long lines from shorter ones. Candidate lines are encoded as a graph where nodes are short coherent lines, and edges are lines that may remain coherent or be merged.

Similar to Shahaf et al. (2015) work, Liu et al. (2017) propose to use a community detection algorithm on a keyword co-occurrence graph to detect topics from a stream of news articles. To extract keywords from a news article to represent the main concepts of the document, the authors combine a gradient boosting decision tree with a logistic regression classifier. The gradient boosting decision tree outputs the raw features and the logistic regression classifier determines whether the extracted feature is a keyword or

not. The study proposes to group similar news articles into events by constructing a keyword co-occurrence graph where edges are added if the times of co-occurrence is above a predefined threshold and the conditional probabilities of the occurrence are above another predefined threshold. Communities in the graph are detected by iteratively trying to utilize the betweenness centrality score (Sayyadi and Raschid (2013)) of edges by removing edges with a high betweenness score until the sub-graph is smaller than a predefined threshold or the maximum betweenness score in the graph is below another predefined threshold. To associate a document with a keyword community, documents and each keyword communities are represented by TD-IDF vectors, and the document is assigned to the keyword community which has the highest cosine similarity. Furthermore, to obtaine fine-grained events, an SVM classifier is trained to determine whether a pair of documents are discussing the same event or not. For each topic, another graph is constructed by assigning edges according to SVN classifier results, then the same community detection algorithm is applied. This sub-graph structure is named as ”story tree” which represents a news story by a tree of events. The authors also propose to run the system in an online fashion by identifying the related story tree upon the arrival of a new event and updating the related story tree. To find the related story tree for a new event, the similarity between the event and existing constructed story trees in the time period are compared and if the similarity is higher than a threshold, the event is added to the story tree by merging with an existing event or extending the story tree. However, selection methodologies of the thresholds used in the study are not mentioned.

In their study, Laban and Hearst (2017) propose a similar method for aggregating large multi-source dataset of news articles into a collection of major stories in an online system and visualizing them to help readers digest the long-ranging stories. In the article, an organizational challenge of collecting news articles is solved by a crawler that works on defined news websites with specific publication date patterns in their URLs formats. The crawled news are extracted and duplicate articles are filtered out by applying a method that groups the extracted articles into small-time ranges (1 week), and creates TF-IDF matrix from the bag of words representations of the articles in a group, then finds the cosine similarity between articles, then apply a defined threshold to find and delete the duplicate articles. In the suggested system, the articles are represented with a set of keywords. Keywords are selected among the words of the article by filtering out the words whose TF-IDF scores are below a defined threshold. With the goal of grouping articles into common local topics, an article graph is constructed from articles over a small-time range by assigning edges between any two nodes if shared common keyword count be-

tween them are above a defined threshold. To create a streamable method were adding new articles updates already existing stories or creates new ones, topic assignment and story detection is run in chronological order using a sliding window with considering three phenomena; linking, splitting, and merging. Linking is the process that compares the current sliding window with the previous by looking at the overlapping articles, and if the majority of nodes in the cluster have previously been assigned to another topic, new articles in the cluster are linked to that topic. Splitting is the phase to handle the cases where one topic is later on separated into two distinct topics. Louvain community detection algorithm is applied to the graph to split the clusters/stories according to community distribution. Merging is the phase to merge stories if a current cluster found contains articles that have already been assigned to two distinct old stories. In the proposed system, to handle the arbitrarily large gaps in time for some stories, for each topic, a vector that keeps counts of keywords is built. If the similarity is above a predefined threshold, and the two topics do not overlap in time significantly, the topics are merged. Similar to Liu et al. (2017), thresholds are selected intuitively by the authors.

Ansah et al. (2019) present a novel graph-based timeline summarization system, named StoryGraph, to track the evolution of the stories from social media sites, such as Twitter. In the study, online Twitter communities are used for the event extraction task. To capture information diffusion between social media users in a community, propagation trees where the nodes are users in a community that is discussing same events of interests and the edges are an adaptation of information between users, are constructed from collected tweets over a specified time period. These social media communities are used to be more confident about the detected topic themes of related events. Given a short-range time window (60 mins), the provocation trees are monitored until a burst threshold is obtained in any tree, and word distribution of the detected event is extracted by tensor decomposition. For the event, an event feature vector that covers community-memberships, word distribution and time, is constructed. To build the story graph, candidate event feature vectors are ordered by their time of the event detected, and for each event, the similarity between the event and previously found topic themes are calculated by Jenson-Shannon divergence and community similarities are measured by Sorensen-Dice coefficient. According to community memberships and text similarities, it is decided to add the candidate event to an existing topic by a node merging operation.

One of the most widely used technique in processing textual data is to represent it using standard TF-IDF weighting scheme. However, TF-IDF might have several limitations like sparsity and missing semantic meanings. In their study, Hu et al. (2017) point

out these deficiencies of TF-IDF and propose a word embeddings based document representation method and an online event detection algorithm. To represent a document with a vector, a method that consists of three steps is applied. Firstly, words embeddings are extracted by skip-gram model. Then, K-means clustering is applied to group words that have similar meanings to obtain a latent semantic space. Finally, a document is represented by the distribution of word cluster by replacing each word in the document with found word clusters indexes. After representing the documents, documents are arranged in chronological order and divided into groups of fixed time slices. Then, the documents in a time slice are processed sequentially, one at a time by applying a threshold on similarities between the document vector and the centroid of the clusters found previously to create a new cluster or join one of the existing clusters. After detecting the clusters, each cluster is compared with each other. If the similarity between two clusters is above a predefined merging threshold, clusters are merged. Finally, to detect and deal with noisy clusters, and variance of the similarities between each document is compared with a predefined noise threshold, and all clusters below the threshold are discarded.

CHAPTER 4

UPDATING NEWS STORIES

In this section, details of the proposed system is explained and definitions for basic concepts are provided to gain the intuition needed to understand the characteristics of a news stream and the inner workings of the proposed model.

The task of story detection can be described as a process of clustering a coherent set of articles that explains a set of related events. Before a clustering algorithm is applied to a document corpus, a mapping of each text document into a compact representation of its content needs to be performed. Selecting a representation method for a document is quite essential for clustering performance as much as the clustering algorithm and the similarity metric used (Lau and Baldwin (2016)).

4.1. Document Representation

To use a clustering algorithm on any form of text, it is required to transform the text into a numeric representation. This numeric representation should exhibit the essential characteristics of the text. The most widely used technique to process textual data is to represent it using standard TF-IDF weighting scheme. TF-IDF scores the importance of a word in a document based on the frequency of the word in that document and the number of documents in the corpus. Despite it is a simple algorithm to transform a text into a vector representation, TF-IDF has several limitations;

(a) High-dimension and semantic sparsity characteristics of the TF-IDF representations might lead high computational costs.

(b) As TF-IDF does not capture the position in the text and semantics, two documents that mention the same topics, but using different words, cannot be correctly handled.

To overcome the limitations of the TF-IDF representations, a typical approach is to develop a document embedding model which produces fixed length vector representations that accurately preserve semantic information within each document. Document embedding can be constructed with supervised and unsupervised algorithms. In this work, we use doc2vec that were proposed by Le and Mikolov (2014) as an extension to word2vec

for learning continuous representations for documents. Doc2vec performs well and robustly when using models trained on a large external corpus (Lau and Baldwin (2016)).

We used the full collection of the Wikipedia article corpus to train our doc2vec model. We firstly clean the Wikipedia article texts by decoding all HTML entities and filtering out wiki markups such as comments, footnotes and URLs. Then, we convert the article texts to lower case, then extract all tokens. After full preprocessing, we ignore articles whose number of tokens is less than 50. After tokenizing the documents, we set up our doc2vec model with *Distributed Bag of Words (DBOW)* model which ignores the word order and predicts probability distribution from words randomly sampled from the paragraph (Le and Mikolov (2014)). Even *DBOW* is a simple model, it produces superior results than another doc2vec embedding model; *Distributed Memory Version of Paragraph Vector (PV-DM)* which predicts a center word from the current context (Lau and Baldwin (2016)). We instantiate *DBOW* model with the found optimal hyperparameters from the work Lau and Baldwin (2016)) which is shown in Table 4.1. Then, the trained model is used to infer document vectors from it. Optimized inference hyperparameters are selected based on the given values in Table 4.2.

Parameter	Description	Value
vector size	”dimensionality of the feature vectors”	300
window size	”maximum distance between the current and predicted word”	15
min count	”parameter to ignore all words with a total frequency lower than this”	20
sub-sampling	”threshold configuring which higher-frequency words are randomly downsampled”	10^{-5}
negative sample	”noise words count that will be drawn”	5
epoch	”the number of iteration”	20

Table 4.1. DBOW training hyperparamaters

Parameter	Description	Value
alpha	”initial learning rate”	.01
min alpha	”minimum learning rate”	.0001
epoch	”number of times to train the new document”	50

Table 4.2. DBOW inference hyperparamaters

The document embeddings that we infer from the doc2vec model have the following advantages:

(a) They capture semantic word relationships. In Table 4.3, two sentences from Your-Dictionary (2019) are given. The sentences are mentioning about "symptoms of the flue" but with using different words. Each sentence is represented by corresponding TF-IDF and doc2vec vectors. Then, we measure the cosine similarity between the vectors. In Table 4.4, cosine similarities between the sentence representations are given. While doc2vec is capable of discovering the similarity between two sentences quite well, TF-IDF does not perform well as it could not capture semantic word relationships, such as semantic relation between *influenza* and *the flu*.

id	sentence
S_1	"Symptoms of influenza include fever and nasal congestion."
S_2	"A stuffy nose and elevated temperature are signs you may have the flu."

Table 4.3. Two sentences that mention about the symptoms of the flue

model	similarity
TF-IDF	0.03
doc2vec	0.68

Table 4.4. Cosine similarities between the TF-IDF and doc2vec based vector representations of the sentences given in Table 4.3

(b) They are learned from unlabeled data in a completely unsupervised way. Thus, it can perform well without needing labeled data.

(c) They can achieve same kind of analogies like word vectors (Dai et al. (2015)). Thus we can perform vector operations on paragraph vectors. We present an experiment in Figure 4.1. To find the article about the *"2016 Nice Truck Attack"*, we perform vector operations on the document vector of *"2016 Berlin Truck Attack"* article. We subtract the word vector of *"Berlin"* from document vector of *"2016 Berlin Truck Attack"*, and add *"Nice"* word vector. Then, we query for finding the top-4 most similar articles from the training set by computing cosine similarities. As it is shown in the Table 4.5, we get the vector of *"2016 Nice Truck Attack"* article as the most similar vector.

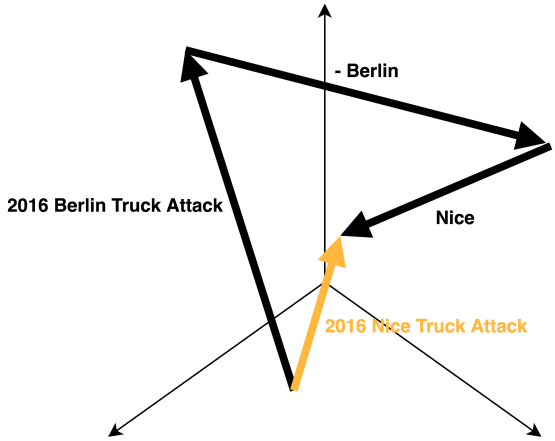

Figure 4.1. Illustration of the vector operation over ”2016 Berlin Truck Attack” article: *D(”2016 Berlin Truck Attack”) - W(”Berlin”) + W(”Nice”)*

article	similarity
”2016 Nice Truck Attack”	**.655**
”2016 Normandy church attack”	**.652**
”Marseille stabbing”	**.642**
”Toulouse and Montauban shootings”	**.631**

Table 4.5. Nearest neighborhoods of the vector operation result given in Figure 4.1

4.2. The Clustering Algorithm

Document clustering is an unsupervised process of dividing a set of documents into groups such that documents belonging to the same groups are more similar to each other in comparison to documents in the other groups. The definition of what constructs a good cluster is not well-defined. Thus, finding out the patterns on which the clustering is performed is the main challenge of the clustering.

Clustering relies on defining a good proximity measure to show the closeness between two data points. There are many proposed similarity or distance measures such as Euclidean distance, cosine similarity, Jaccard coefficient, etc. In this work, we compute the similarity between two data points with cosine similarity. Cosine similarity is calculated by the cosine angle between two vectors by the equation given in Equation 4.1.

$$cos(\boldsymbol{d}_1, \boldsymbol{d}_2) = \frac{\boldsymbol{d}_1 \cdot \boldsymbol{d}_2}{||\boldsymbol{d}_1|| \cdot ||\boldsymbol{d}_2||} \tag{4.1}$$

where d_1 and d_2 are document vectors.

Cosine similarity is one of the most popular similarity measures applied to text document. It also performs quite well for the clustering tasks (Lau and Baldwin (2016)).

A graph data structure can easily capture the relationships of documents. A common property of the networks is that some set of nodes are more densely connected and interact more with each other than with nodes outside of the set. In social networks, for instance, people in a certain community interact with each other much more frequently than they do with the people in the other communities(Girvan and Newman (2002)).

In this work, we propose to represent news article relations with a network. In the news article network, certain articles will likely have more conceptual overlap each other than the others, and intuitively we expect this type of dense article communities to represent news stories.

We represent each news article as a node in a fully connected undirected weighted network, the edge weights between the nodes are set based on the cosine similarities between the document vectors.

id	News Articles
0	”Nice: Bastille Day Attack - Timeline of Events”
1	”Nice attack: At least 84 killed by lorry at Bastille Day celebrations”
2	”Attack in Nice: Truck driver 'radicalized quickly”
3	”Turkey's Erdogan always feared a coup. He was proved right.”
4	”Manchester United transfer rumours: Cesc Fabregas and Paul Pogba linked”
5	”Nice Truck Attack: Five Detained as Grief Turns to Anger”
6	”The coup attempt is bad news for Turkey's democracy”
7	”Turkey coup: how defiant Turks chased soldiers from Bosphorus Bridge”

Table 4.6. Sample news articles

To detect the news stories, it is essential to understand the overall functioning of the network by finding out the underlying community structure, and hierarchical levels of the communities in the network. Thus, we propose to use a community detection algorithm to find out the stories and their hierarchies. To illustrate the process, we set up a sample fully connected undirected weighted network from the news articles collected from Google News, shown in Table 4.6. Figure 4.2 visualises the network of the news articles.

Many community detection algorithms have been proposed to understand the underlying structure of a complex network and extract communities, such as Louvain (Blon-

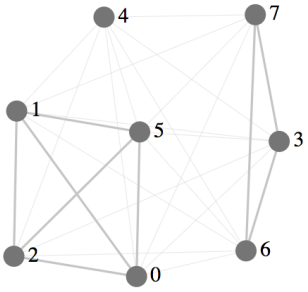

Figure 4.2. News article network of the articles given in Table 4.6

del et al. (2008)), Fast Greedy (Girvan and Newman (2002)), MarkovCluster (van Dongen (2008)), and InfoMap (Rosvall and Bergstrom (2008)). To detect the stories, we use the Louvain method. The Louvain community detection method is an algorithm for finding communities in the network by trying to maximize the modularity in a repeated process. The Louvain method runs faster in comparison to preexisting algorithms and it can work well on much larger networks (Blondel et al. (2008)). One of the major advantages of the Louvain method, it reveals a hierarchy of the communities at different scales, so for the news story point of view, this hierarchical perspective helps us understand the story construction and observe root stories. Figure 4.3 shows the detected communities on the news articles network given in Figure 4.2.

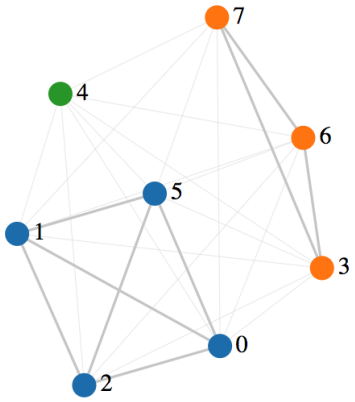

Figure 4.3. Detected communities on the news articles network defined in Figure 4.2

4.3. Story Representation

Learned document representations capture meaningful semantic regularities and relations, and this allows us to apply simple algebraic operations on the document vectors as seen in Figure 4.1. Thus, in this work, we propose to use this impressive feature of the document embeddings to form document relations and come up with story representations.

As we describe in the clustering algorithm section, we find communities in the document network using a community detection algorithm. A community is a set of data points that are similar to each other and relatively dissimilar with the data points in other communities (Girvan and Newman (2002)). Our idea is that a community in a news article network can be considered as a story which consist of news articles that are about the related events.

id	News Articles
1	”Turkey’s Erdogan always feared a coup. He was proved right.”
2	”The coup attempt is bad news for Turkey’s democracy”
3	”Turkey coup: how defiant Turks chased soldiers from Bosphorus Bridge”

Table 4.7. Sample articles belonging *Turkey coup* story

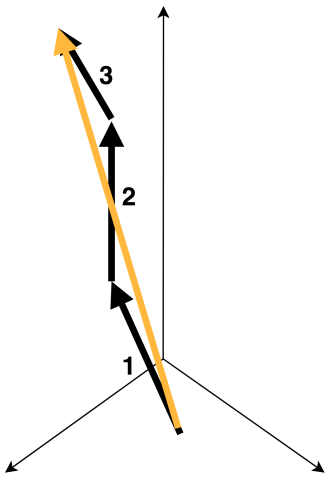

Figure 4.4. *Turkey coup* story vector is obtained by adding up document vectors assigned to that story.

Accurately representing a story with a numeric vector is essential to discover the

underlying relations and similarities among news stories. Thus, we propose to represent a story with the semantics of the documents that belong to it. As it is shown in Figure 4.4, a story can be represented as an embedding vector by adding up all the document vectors in the story to give a total displacement vector from the initial point of the first news article vector to the terminal point of the last news article vector inside the story. We can also represent the communities found in Figure 4.3 as stories. This execution is shown in Figure 4.5.

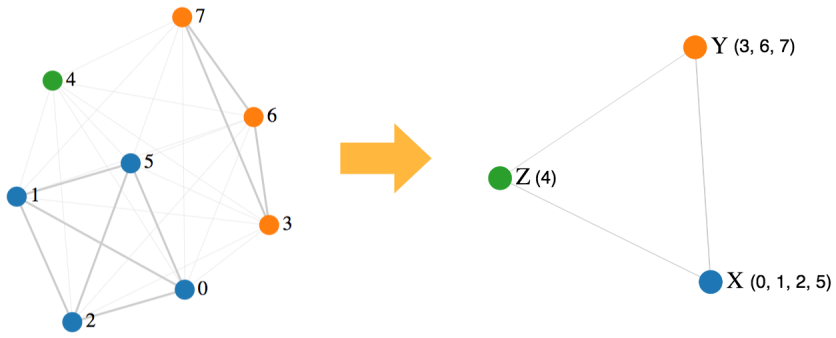

Figure 4.5. All the documents belonging to same community found in Figure 4.3 is added to represent related stories

4.4. Online Clustering

One of the main challenges of clustering is that many datasets are large and can hardly fit into memory. Furthermore, while the amount of data grows, generally traditional clustering algorithms that are primarily optimized for the accuracy becomes intractable in terms of computational time and detecting accurate clusters out of the dataset becomes a challenge (Kurasova et al. (2014)). To overcome this issue, stream clustering algorithms are required for extracting useful knowledge from data streams.

An ideal data stream clustering algorithm should fulfill the following requirements (de Andrade Silva et al. (2013)):

(a) perform the clustering incrementally in a tractable way.

(b) quickly adapt to the dynamics of the streaming data.

(c) scale well as the amount of data grows continuously.

(d) data representations should not grow with the number of data processed.

(e) detect the outliers and act accordingly.

(f) work with different data types.

In practice, these requirements are only partially fulfilled and optimizing the stream clustering approach according to the given problem definition is also quite common (de Andrade Silva et al. (2013)).

One common technique for online clustering is evaluating the new data point only over sliding windows of recent data. For instance, only data from the last week can be used to cluster new data instead of querying over entire past history. Sliding window approach emphasizes only the recent data, and for many applications, recent data are more important and relevant than the old one.

Majority of the news articles belonging to the same story are published in a dense way in a short period of time. Thus, feeding news in the chronological order to the system and using a sliding window approach to group and process data, produces informative and useful semantics to detect news stories.

Many existing works on online news story detection feed the news stream in the chronological order to their systems and use sliding window method, and time-sensitive queries to cluster the documents (de Andrade Silva et al. (2013)). In such systems, defining an optimal sliding window interval is a challenge. Setting a short sliding interval like a minute might cause computational challenges, and setting a relatively longer interval might cause to miss useful semantics of the recent data. Thus, introducing new approaches that address this issue is important.

4.4.1. Inching Window

Generally, the main concern of a data streaming system is processing new data elements on-the-fly, but processing according to a time interval of the data requires gathering the data until interval condition is met. For instance, a sliding window of five days with a one-day sliding interval requires to wait until the end of the last day in the window to gather all data. To overcome this issue, we propose a novel sliding window technique, named *"Inching Window"* as the window slides like an inchworm. Inchworms move in a characteristic *inching* or *looping* gait by extending the front part of the body and bringing the rear up to meet it" (MerriamWebster (2019)). An *inching window* moves in a similar

fashion. Given a window size of 3 days, and a sliding interval of 1 day, similar to sliding window, *inching window* groups the incoming data in batches every 3 day, but rather than sliding directly after processing the batch, it continues to process each incoming data one by one until the sliding interval (1 day), then it slides. This *inching window* process is illustrated in Figure 4.6.

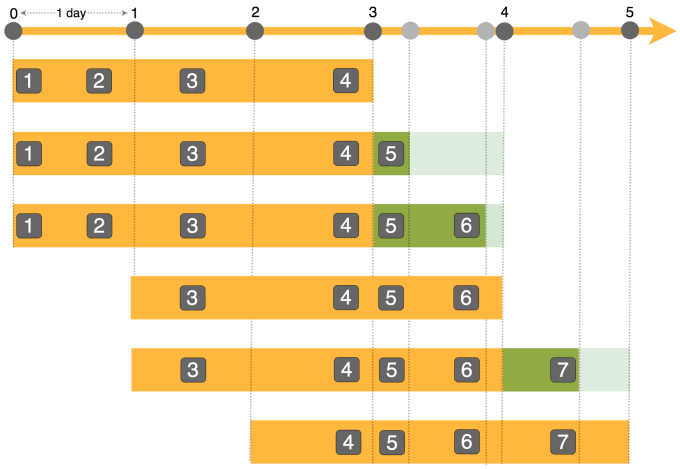

Figure 4.6. A sample inching window setup

The main advantage of the *inching window* approach over *sliding window* is that *inching window* enables processing of the new data elements on-the-fly within the recent data context.

After feeding a news stream in the chronological order to an *inching window* set up, we propose an online story detection system that is composed of three main steps; topic creation, on-the-fly document clustering, and story construction.

4.4.2. Topic Creation

A set of news articles in a small time interval (e.g. 5 days) have a potential to create a densely-connected local group around a *topic* which, can then be merged in a *story*. To detect the topics in a window, we initiate a news article network and apply the clustering algorithm proposed in Section 4.2. The clustering algorithm groups densely-connected news articles in the window into communities that we call as topics. Topics can be represented with document embeddings by adding all document vectors belonging to the topic in a similar faction described in Section 4.3.

4.4.3. On-The-Fly Document Clustering

After the initial inching window, news articles are received one-by-one until the specified window interval. This enables us to process a newcomer article without waiting for the end of the day. However, it is not computationally feasible to apply the proposed clustering algorithm again and again for each newcomer article. Furthermore, it is often acceptable to assign a newcomer article to the most suitable community in the given context, and later on, review the temporary assignments and make permanent assignments.

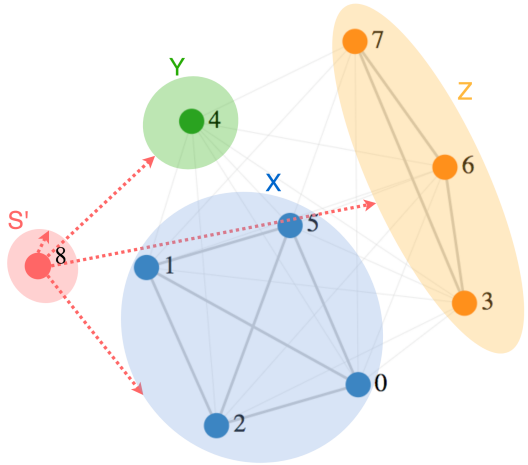

Figure 4.7. On-the-fly clustering process

The clustering algorithm that we propose in Section 4.2, relies on Louvain community detection method that tries to identify communities in a network by repeating optimization steps iteratively until a maximum of modularity is attained. Modularity is designed to measure the strength of division of a network into communities (Girvan and Newman (2002)). To assign a new document into a community in a fast and memory-efficient manner, we propose to compare modularity changes in the network after assigning the new document into one of the existing communities or creating a new community for the document. The on-the-fly news clustering process is illustrated in Figure 4.7. Given a network with communities found in Figure 4.3, for the newcomer $Article_8$, we measure modularities on the graph for the scenarios where $Article_8$; joins $Community_X$, joins $Community_Y$, joins $Community_Z$, creates a new community, named $Community'_S$. Then, we select the scenario that the graph modularity is maximized. For the given sample, we found that community X maximizes the modular-

ity, so $Article_8$ is assigned to $Community_X$.

We used the topic creation process that is described in Section 4.4.2 as the initiator and reviewer of the on-the-fly clustering. Before sliding a window, we apply the *Topic Creation* process all over the window and override the temporary assignments.

4.4.4. Story Construction

A *topic* that we detect might be; 1. an initial seed of a new story, 2. belonging to an active story, 3. a continuation of a story that is interrupted in time, for instance, a crucial piece of evidence of *Malaysia Airlines Flight 370 Disappearance* are found after years.

To properly address these characteristics of a topic, we propose to create another fully connected network where the nodes are stories detected, and the edges' weights are cosine similarities between the nodes. First, we represent topics found in a window with the sum of the vectors representation of the articles belonging to that topic. Then, we added the topics as a node to the story network that we initiated. Then, we apply the community detection algorithm that we proposed in Section 4.2. Running community detection over the story network creates interesting dynamics:

(a) A topic might create a community with an existing story in the network. In this case, we merge the topic with the story, then remove the topic node from the network. Merging is executed by summing up the story vector and the topic vector after handing the document migrations from one story to another. Sliding windows overlap in time and data. Thus, a topic found in a window can have the same document with a story found in the previous window. Thus, when a story and topic is merged, if a document is already belonging to another story, it should be removed from that story first.

(b) A topic might create a new individual community. After handling the document migrations, the topic is directly casted into a story.

(c) A topic might create a community with another topic found in the same window. Communities are decided based on the modularity of the network. Two document communities found in the local article network context might be merged into one in the global story network. Merging is executed in the same fashion with topic-story merging. After the merging topics,

(d) An existing story might create a community with another story. This case rarely happens, but as the stories grow in time, their relations with each other evolve as well. Thus, two distinct stories might start to have connections and finally be merged. Two stories are merged on the oldest one by directly summing up their vectors.

Stories do not need to keep individual document data, creating a story from a topic, keeping only the documents ids and sum of the document vectors to represent the story is enough. Documents data and their individual vector representations can be deleted from the memory.

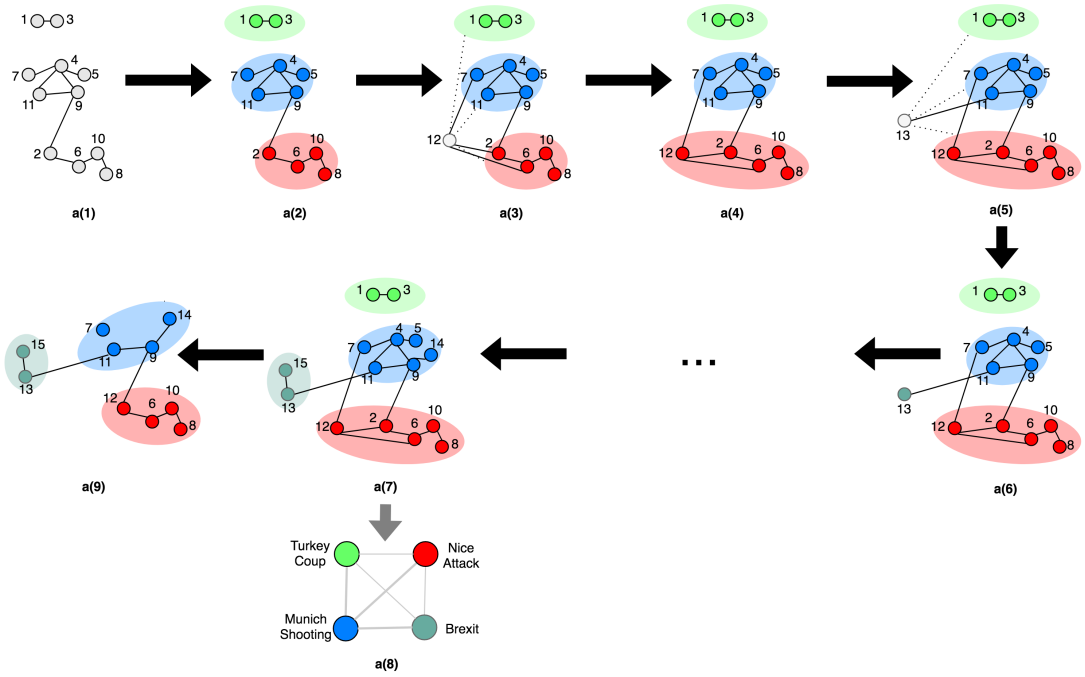

Figure 4.8. A sample execution of the proposed algorithm

To illustrate the process, we executed our proposed method in a sample mini-corpus of 15 news articles between 24.06.2016 and 29.06.2016, listed in Table 4.8. The corpus consists of news articles on three stories; *Turkey's Coup*, *Nice Attack*, *Munich Shooting* and *Brexit*. We set up an inching window for 4 days with the inching interval of 1 day.

(a) Subfigure a(1) shows the network of the first window articles between 24.06.2016 and 28.06.2016.

(b) In the Subfigure a(2), the clustering algorithm proposed in Section 4.2 is run to detect the topics.

(c) Forward moving process of the inching window is started in Subfigure a(3). To assign the newcomer $Article_{12}$ to a community, the graph modularity is iteratively checked for each possible community assignments.

(d) In the Subfigure a(4), $Article_{12}$ is assigned to the community that produces the maximum modularity.

(e) Subfigure a(5) and Subfigure a(6) applies the same procedure with the Subfigure a(3) and Subfigure a(4) for the $Article_{13}$, but this time, creating a new community maximize the graph modularity.

(f) On-the-fly clustering is executed for each new document until $Article_{15}$ which is the latest article of the window. This last step of the window is shown in Subfigure a(7)

(g) Topic clustering algorithm run again to review and correct the on-the-fly community assignment in Subfigure a(8).

(h) In the Subfigure a(8), the final set of topics are added to the story network and proposed flow for the story construction is executed.

(i) In the Subfigure a(9), we shrink the inching window by removing the first day of the window; 24.06.2016.

ID	Date	Source	News Title
1	24.06.2016	middleeasteye.net	”Turkey prepares for post-coup unity rally”
2	24.06.2016	dailymail.co.uk	”The multiple lives of Nice attacker”
3	24.06.2016	reuters.com	”Turkish president gains upper hand in power struggle”
4	24.06.2016	cnn.com	”Munich gunman planned attack for a year, officials say”
5	24.06.2016	wsj.com	”Terror Attacks Weigh on Europe’s Travel Companies ”
6	25.06.2016	washingtonpost.com	”Scandal grows over lack of security on night of Nice terror attack”
7	25.06.2016	time.com	”German Police Arrest Afghan Teen in Connection With Munich Shooting”
8	26.06.2016	cnn.com	”Two more people arrested in connection with attack in Nice”
9	26.06.2016	wsj.com	”Germany faces one of its greatest political challenges since World War II”
10	27.06.2016	businessinsider.com	”Flight bookings for Nice and France drop again after Bastille Day terror attack”
11	28.06.2016	theguardian.com	”Munich gunman saw sharing Hitler’s birthday as ’special honour’”
12	29.06.2016	ft.com	”French tourism falls victim to terror attacks”
13	29.06.2016	mashable.com	”Stephen Hawking has a stark warning for what Brexit could mean for the human species”.
14	29.06.2016	alternet.org	”Munich Shooter Considered Himself Aryan, Admired Hitler and Breivik”
15	29.06.2016	independent.co.uk	”Stephen Hawking warns human race will perish if pre-Brexit attitudes towards money are not challenged”

Table 4.8. A mini-corpus of 15 news articles between 24.06.2016 and 29.06.2016

CHAPTER 5

EXPERIMENTAL RESULTS

In this section, we describe the experimental results of the proposed news clustering algorithm. To evaluate the performance, we construct an experimental setup to address both accuracy and reliability.

In the experiments, we used two different metrics;

(a) **F1**: F1 score is an overall measure of a model's accuracy which is calculated by the harmonic mean of the precision and recall scores (Equation 5.1). Precision is a measure of relevancy that refers to the percentage of positive identifications that are actually correct, while recall is a measure of sensitivity that refers to the percentage of actual positives that are identified correctly.

$$F1 = \frac{Precision \times Recall}{Precision + Recall} \tag{5.1}$$

A high F1 score indicates a low rate of false-positives and low rate of false-negative. If F1 score of a model is 1, then it is considered as perfectly clustered, and if F1 score of a model is 0, it indicates that the model results are totally wrong.

(b) **Normalized Mutual Info Score**: Normalized mutual information (NMI) is a popular metric for performance evaluation of community detection algorithms. Normalized mutual information is built on the entropy concept in information theory (Shannon (2001)). For given two random variables, X_1 and X_2, NMI between X_1 and X_2 is computed by the given equation in Equation 5.2

$$NMI(X_1, X_2) = \frac{I(X_1, X_2)}{\sqrt{H(X_1) \times H(X_2)}} \tag{5.2}$$

where $I(X_1, X_2)$ is the mutual information between X_1 and X_2. $H(X_1)$ and $H(X_2)$ are the entropies of X_1 and X_2, respectively.

The closer NMI score is to one, the better the quality.

In our first experimental setup, we would like to see the performance of the proposed methods on short-ranging story tracking, long-ranging story tracking and tracking stories that are interrupted in time.

Date	News Search Terms (with article count)
15.07.2016	Nice (28), Coup (16), Paul Pogba (10)
16.07.2016	Nice (22), Coup (26), Hillary Clinton (11), Nate Thurmond (7)
17.07.2016	Nice (21), Brexit (3)
18.07.2016	Nice (20)
19.07.2016	Nice (15), Presidential Election (12)
20.07.2016	Nice (16), Donald Trump (25), Melania Trump Speechwriter (17)
21.07.2016	Brexit (16)
22.07.2016	Nice (13), Munich (13)
23.07.2016	Munich (19), Brexit (8)
24.07.2016	Munich (19), Brexit (9), Coup (21)

Table 5.1. A scenario for performance evaluation

To evaluate the performance on these items, we follow a scenario with the stories shown in Table 5.1.

For the construction of the annotated news article corpus that we use as the ground-truth, we used Google News as our news article provider. We wrote a custom crawler that downloads all first page articles according to a given keyword and a time-range specification. Using the crawler, we download all first-page news articles between 15.7.2016 and 24.7.2016 with given search keywords; *"Nice"*, *"Coup"*, *"Paul Pogba"*, *"Hillary Clinton"*, *"Nate Thurmond"*, *"Brexit"*, and *"Munich"*.

In the scenario that we follow, we used *"Nate Thurmond Dies"* story to observe short-ranging story tracking performance. *"Nice Truck Attack*" story to observe long-ranging story tracking performance. *"Turkey Coup"* story to observe the performance on tracking stories that are interrupted in time.

We would also like to see the performance of the proposed system on differentiating two similar events happened around the same time, but in different location. For this experiment, we used the *"Munich Attack"* story that is happened around the same time with *"Nice truck Attack"* story.

Furthermore, we used the *"US Presidential Election"* story to evaluate the performance of the clustering the news articles on the same topic, but with differences in time, places and the people involved. For this evaluation, we expect to cluster news articles about *"Donald Trump"*, *"Hillary Clinton"*, and *"Melania Trump"* election performances, into one.

ID	Base Topic	Cluster Size	Keywords (with number of occurrences)
$Cluster_1$	”Nice Attack”	101	nice (94), attack (66), truck (18), terror(17)
$Cluster_2$	”Nice Attack”	30	nice (25), attack (24), student (17), berkeley (11)
$Cluster_3$	”Nice Attack”	2	nice (2), attack (2), rhode (2), mosque (2)
$Cluster_4$	”US Election”	67	trump (48), donald (18), melania (17), clinton (15)
$Cluster_5$	”Turkey Coup”	61	coup (51), turkey (47), attempt (17), turkish (13)
$Cluster_6$	”Pogba Transfer”	11	united (10), manchester (9), pogba (8), paul (8))
$Cluster_7$	”Nate Thurmond Dies”	7	nate (7), thurmond (7), warriors (6), hall (5)
$Cluster_8$	”Brexit”	37	brexit (31), uk (6), british (5), says (5)
$Cluster_9$	”Munich Attack”	50	munich (43), shooting (17), attack (12), gunman (10)

Table 5.2. Clusters after running the proposed clustering algorithm

We feed the corpus as a stream to the proposed clustering method. Table 5.2 shows the stories found with the number of news articles belonging to the stories. *Nice Attack* story is divided into three. To observe the difference between these clusters, we extracted most common 4 keywords from article headlines for each cluster. We saw that while $Cluster_1$ is the main story for the Nice truck attack, $Cluster_2$ gathers news articles about efforts to find a UC Berkeley student missing in terrorist attack in Nice, and $Cluster_3$ collects two news articles about a mosque in Rhode Island that was vandalized after the Nice attacks. According to definition and scope of the news story, this separation might be accepted as relevant in common sense. From Table 5.3, the performance results can be seen. We achieve to get high F1 and NMI scores that indicates good clustering accuracy and quality.

F1	NMI
0.829	0.823

Table 5.3. The clustering results compared to the ground-truth

CHAPTER 6

CONCLUSIONS

In this work, we present a method for aggregating news articles into topics and then merge them into story clusters in online fashion from a continuous stream.

Main challenges of extracting the topics are as follows; representing the documents in the dataset, finding out the similarities between the documents, and clustering the similar documents into a group. Many traditional studies on text clustering rely on TF-IDF as document representation model. It might have several limitations such as sparsity, computational cost and missing semantic relations of the words. To overcome these limitations, we propose to use doc2vec(Le and Mikolov (2014)) for news article representations. Relations between the articles can be modeled in a network structure. Similar documents have strong connections with each other and have the potential to represent a topic. To address this behavior, we propose to use the Louvain community detection algorithm for grouping similar articles as a topic. Many datasets are large and can hardly fit into memory, thus, it is essential to process data in online fashion. In this space, we present a modified sliding window approach called *"Inching Window"*. Online clustering consists of two steps; on-the-fly clustering where each newcomer article is directly assigned to a community that maximizes the graph modularity, and story construction process where the detected topics are merged into stories. Finally, we download a set of news articles on specific stories and use this corpus to validate the performance of the algorithm. The results show that the algorithm is effective to discover meaningful stories from the given corpus.

6.1. Future Work

Revealing communities in document article network allows for analysis of relations between individual articles and classifications of the objects based on their characteristics. In some real-life networks such as social networks, objects can simultaneously belong to multiple communities at once. These communities are called overlapping communities. In this study, we follow the general traditional sense that a news article can belong to only one news story at a time. However, analysis of the overlapping news sto-

ries might create interesting dynamics. We plan to change switch Louvain community detection algorithm with an overlapping community detection algorithm and evaluate the performance of the system.

Although the focus of the work is to find stories from the news articles, we plan to test the proposed system with different types of data, such as social media news.

In this work, we evaluate the accuracy with a manually collected document set. Real word data might have more noise and different characteristics. We plan to evaluate the accuracy of the story creation algorithm in a large annotated corpus.

REFERENCES

Allan, J. (2002). Topic detection and tracking: event-based information organization.

Ansah, J., L. Liu, W. Kang, S. Kwashie, J. Li, and J. Li (2019). A graph is worth a thousand words: Telling event stories using timeline summarization graphs. In *WWW*.

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). Fast unfolding of communities in large networks.

Can, F., S. Kocberber, O. Baglioglu, S. Kardas, H. C. Ocalan, and E. Uyar (2010). New event detection and topic tracking in turkish. *JASIST 61*, 802–819.

Choudhary, R., S. Mehta, and A. Bagchi (2008). On quantifying changes in temporally evolving dataset. In *CIKM*.

Dai, A. M., C. Olah, and Q. V. Le (2015). Document embedding with paragraph vectors. *CoRR abs/1507.07998*.

de Andrade Silva, J., E. R. de Faria, R. C. Barros, E. R. Hruschka, A. C. P. de Leon Ferreira de Carvalho, and J. Gama (2013). Data stream clustering: A survey. *ACM Comput. Surv. 46*, 13:1–13:31.

Girvan, M. and M. Newman (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America 99 12*, 7821–6.

Goyal, R., R. Malla, A. Bagchi, S. Mehta, and M. Ramanath (2013). Esthete: a news browsing system to visualize the context and evolution of news stories. In *CIKM*.

Hu, L., B. Zhang, L. Hou, and J.-Z. Li (2017). Adaptive online event detection in news streams. *Knowl.-Based Syst. 138*, 105–112.

Jones, K. S. (1988). A statistical interpretation of term specificity and its application in

retrieval. *Journal of Documentation 60*, 493–502.

Kurasova, O., V. Marcinkevicius, V. Medvedev, A. Rapecka, and P. Stefanovic (2014). Strategies for big data clustering. *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, 740–747.

Laban, P. and M. A. Hearst (2017). newslens: building and visualizing long-ranging news stories. In *NEWS@ACL*.

Lau, J. H. and T. Baldwin (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Rep4NLP@ACL*.

Le, Q. V. and T. Mikolov (2014). Distributed representations of sentences and documents. In *ICML*.

Liu, B., D. Niu, K. Lai, L. Kong, and Y. Xu (2017). Growing story forest online from massive breaking news. *ArXiv abs/1803.00189*.

MerriamWebster (2019 (accessed May 30, 2019)). *Inchworm*. `https://www.merriam-webster.com/dictionary/inchworm`.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *NIPS*.

OpenCalais (2019 (accessed May 30, 2019)). *OpenCalais API*. `http://opencalais.com`.

Rosvall, M. and C. T. Bergstrom (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America 105 4*, 1118–23.

Sayyadi, H. and L. Raschid (2013). A graph analytical approach for topic detection. *ACM Trans. Internet Techn. 13*, 4:1–4:23.

Shahaf, D., C. Guestrin, E. Horvitz, and J. Leskovec (2015). Information cartography. *Commun. ACM 58*, 62–73.

Shannon, C. E. (2001, January). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev. 5*(1), 3–55.

van Dongen, S. (2008). Graph clustering via a discrete uncoupling process. *SIAM J. Matrix Analysis Applications 30*, 121–141.

YAGO (2019 (accessed May 30, 2019)). *YAGO*. `http://www.mpi-inf.mpg.de/yago-naga/yago/`.

YourDictionary (2014 (accessed May 30, 2019)). *Examples of Paraphrasing*. `https://examples.yourdictionary.com/examples-of-paraphrasing.html`.