Bilkent University

Department of Computer Engineering

Senior Design Project

Newspector: A Reliable Platform for News

Final Report

Ahmet Ayrancıoğlu	21601206
Deniz Dalkılıç	21601896
Kaan Gönç	21602670

Supervisor: Varol Akman

Jury Members: Halil Altay Güvenir and Özcan Öztürk

Final Report

May 27, 2020

1. Introduction

The news is one of the most important parts of everyone’s life as it is a medium for people to learn about the latest developments in the world. The news directly affects people and society as a whole. No matter where you are in the world either traveling or working in the office you can get all the latest updates from the world. Without the circulation of the news, it is impossible for people to know what is happening around them. The latest news covers all the news from various fields such as sports, entertainment, fashion, charity events, and much more. The world is changing every moment and, we can keep an eye on the changes through the news. Therefore, the importance of the news in our daily lives is immeasurable. This important role of news in the world lead to various news companies to be founded and thus creating a great rivalry between such sources on being the fastest and the most reliable one among all to deliver news to society. However, with every important part of humanity, there are also important issues that come with it.

The first issue is “Fake News”, as people are concerned about the presence of fake news [1] all around the world for several reasons. The term “Fake News” is interpreted as “false news stories, often of a sensational nature, created to be widely shared or distributed for the purpose of generating revenue, or promoting or discrediting a public figure, political movement, company”, according to the Oxford Dictionary. Spreading of fake news is a real and extremely serious threat in today's society as it can lead to violations of rights to information and expression. This in return can be used as a tool of manipulation in terms of prejudice, marginalization, excessive assimilation of ideas, acts, people, products, or organizations. Furthermore, there are excessive amounts of fake news present on the internet. The main reasons for this are the ease of writing/authoring fake news; newspapers, magazines, and news agencies having their online channels, and social media being an effective medium for transmission. The internet and social media have made it very easy for anyone to publish content on a website, blog, or social media profile and potentially reach large audiences. As most of the people who are reading news is getting the material from social media, many content creators use this as an advantage. These kinds of news can be encountered daily in various fields such as politics, economics, social life, security, humanity. Most importantly, it is extremely hard to prevent them for 3 main reasons. Firstly, it is difficult to identify the source of the news. Secondly, it takes too much human effort and time to crosscheck and confirm their reliability. Finally, they instantly start to spread around the internet when they are published.

The second issue is the variety of perspectives. When we surf on the internet we are exposed to news, articles, and other content based on our own routines on the web. Such content tends to reflect our own likes, views, and beliefs and therefore isolating us from differing views and opinions present on the web. Additionally, sometimes these two problems are combined and people perceive different perspectives as fake simply because it contradicts their ideas and views.

The final issue is the competition between news sources. With this many news sources, there come some caveats such as which news source is the most reliable or which news source someone should follow in order to access the latest developments around the world first.

By looking at all these issues together, we can see that the concept of accessing reliable breaking news comes out to be a real concern for many people. Anything may happen at any point in time, so it is very important for people that the media stay alert to cover the incidents when it happens.

In order to tackle all these problems together, exposing people to multiple versions of the same story from different perspectives and sources proves crucial. By simply showing different contents from various perspectives together, we can increase the ability of critical thinking of the readers. And by leaving the judgment and the choices to the reader without the influence of other people's rights or wrongs, we can provide a safe platform for people to consume news with resources that would help them decide whether a news story is reliable or not on their own. We are planning to define worldwide known and trusted news companies' websites and twitter accounts as our sources of data in our project and create this safe platform for people to consume news stories.

2. Requirements Details

2.1 Functional Requirements

Frontend Applications

- The Newspector application should be able to display the newsgroups in sorted order.
- The Newspector application should be able to display the news stories in a newsgroup in chronological order inside a timeline.
- The Newspector application should be able to display a news story with a detailed analysis of the story.
- The Newspector application should provide access to the tweet and the webpage of the news story.
- The Newspector application should be able to display newsgroups filtered by different categories.
- The user should be able to follow/unfollow newsgroups.
- The user should get notifications when a news story is added to a newsgroup that they are following.
- The user should be able to tap on the notification to go to the newsgroup page.
- The Newspector application should be able to display the newsgroups followed by the user.
- The Newspector application should be able to display all the news sources used to gather the news story.
- The Newspector application should be able to display statistics about the news source and comparisons between the news sources.
- The Newspector application should provide access to the twitter page and the webpage of the news source.
- The user should be able to rate the news source at most once a day.
- The user should be able to report news stories.

- The user should be able to sign in to the Newspector application using their google accounts.

NLP Server Application

- The application should be able to group related news in the same clusters and newsgroups.
- The application shouldn't group unrelated news in the same clusters and newsgroups.
- The application should be able to assign the new-coming news to the proper clusters groups.
- The application should be able to create a new cluster and new newsgroup when no proper cluster is found for the new-coming news and it should assign that news to the created cluster and newsgroup.
- The application should be able to create a new newsgroup for the cluster when a proper cluster is found for the new-coming news but there is no active newsgroup in that cluster. Then, the application should assign that news to the created newsgroup.
- The application should be able to perform batch clustering frequently for the recent news.
- The application should be able to detect that a piece of new-coming news is fetched and process it as soon as possible.

Information Retrieval Server Application

- The application should be able to fetch the latest account information of a Twitter account.
- The application should be able to fetch the latest posts from a Twitter account.
- The application should be able to update the last time a post is fetched for each Twitter account.
- The application should be able to perform the categorization of the content of a post.
- The application should be able to perform a sentimental analysis of the content of a post.
- The application should be able to filter the posts that include breaking news with relevant tags.

- The application should be able to remove any unnecessary words or characters from the content of the post.

- The application should be able to identify any important label that a post has such as "Watch Now", "Update" and etc. which are related to the breaking news concepts.

- The application should be able to upload the account information fetched to Google's Firestore Database.

- The application should be able to upload the posts fetched to Google's Firestore Database.

Google Firebase Cloud Application

- The application should be able to send notifications to users who follow a newsgroup when a new article is added to that newsgroup.

- The application should be able to update the count of likes of an account in Google's Realtime Database.

- The application should be able to update the count of dislikes of an account in Google's Realtime Database.

- The application should be able to update the count of reports of an account in Google's Realtime Database.

- The application should be able to update the count of reports of news in Google's Realtime Database.

2.2 Non-Functional Requirements

Supportability

- All classes and scripts should have a description explaining the functionality of the class and the reasoning behind its creation. Additionally, all functions should also have brief descriptions that give insight into why the developer chose to write that function.

- The project should be divided into well-defined and independent subprojects and layers in order for multiple people to work on the overall application together efficiently.

- The project should be structured in a way that would support extending the project even though some of the decisions for achieving said extendibility could be considered over-engineering by some for the initial state of the project.

Usability

- The Newspector application should be available to download in the Google Play Store and Apple App Store.

- The Newspector application UI should follow both Apple's Human Interface Guidelines [2] and Google's Material Guidelines [3].

- The Newspector application should use icons and images to help make information more understandable.

- The notifications sent out by the Newspector application should make noise and display visual alerts.

Reliability

- Gathering the news pieces from selected sources is crucial for the Newspector to function. So, Newspector should be able to gather news from at least 90% of the selected sources at a given time.

- The sources should be selected carefully by a knowledgeable person in order to reduce bias.

- Newspector should update its sources so that no deadlinks will ever be presented to a user such as removed news articles, etc.

- The data storage solution for the project should be selected in a way that will allow the Newspector to be functional when a crash happens on one of the servers that store the news data.

- The data storage solution for the project should be selected in a way that will support evergrowing news stories without being overwhelmed and causing a crash.

Performance

- The gathering of the news from selected sources should be fast and effective to include the most up to date news.

- The servers running the clustering algorithms should be fast and can be auto-scaled as the population of the userbase grows to provide the same performance for every user.

- The fetching the newsgroups, news stories and news sources from the Newspector databases should be fast. Newsgroups should be fetched from the database in less than 1 second for 5 newsgroups.

- The animations, scrolling, navigation and etc. should be smooth. The Newspector application should not drop down from 60fps.

- The authentication and sign-in process should not take more than 2 seconds.

- The notifications should reach the user within 10 minutes after being sent from the servers.

- The application should be up and operational in less than 2 seconds after opening the application after the killing of the application.

Extendibility

- The addition of further news sources should be easy. Other parts of the application should not be dependent on where the news is coming from.

- The addition of further categories should not affect the rest of the projects including the application and the servers.

- Adding new filtering and sorting options to newsgroups should be easy and only require changes to the queries made by the application, not other systems.

Portability

- The process of porting the Newspector to other mobile or web platforms should be easy and should be connected to the same backend server.

- The user experience of the Newspector should be the same on various platforms.

Privacy

- Data gathered by users should not be associated with users directly unless they give their consent.

- Data gathered should be securely stored in our databases as it can easily be used to target news or advertisement by other platforms.

3. Final Architecture and Design Details

3.1 Overview

The Newspector uses Natural Language Processing, Text Analysis, and Sentiment Analysis algorithms to examine and make comparisons of news headlines. The platform consists of two main layers. The first one is the frontend applications that users can interact with. These applications are currently an android application and an ios application, with a web application being possible with enough demand. The second layer is the backend applications that execute all the gathering, examination analysis, and comparison processes mentioned before. There are multiple backend applications for various tasks that will be detailedly explained in the following parts of this report.

There are 2 main ways that backend and frontend applications communicate with each other. The first way is indirect communication using the database. Backend applications modify the database and front-end applications listens to there changes and react to them. Or the opposite in which the front-end applications change the database and backend applications react to the changes. The second way is direct communication. This way of communication only happens while sending a notification to the front-end applications when a new story is added to our databases. This direct communication is necessary as notifications should arrive at the front-end applications even when the application is closed.

In order to achieve a great user experience, we used a software architecture design pattern that helped us with the implementation. As we mentioned previously, the Newspector consists of the front-end applications and multiple backend applications. We decided to use a software architecture design similar to Model-View-Controller (MVC) [4] for our user interface applications in order to create a codebase that is reusable, easy to modify and safe to use.

We developed our front-end applications using the Flutter [5] framework and the Dart [6] programming language and we use Google's Firestore [7] as our database, which is a NoSQL database which is flexible, reliable, fast, and scalable. Information about news stories, newsgroups, and news sources are kept in the Firestore database and these entities are paralleled by the models in the frontend applications. Newsgathering, categorization, clustering, analysis, etc. are done by the back-end applications using the entities kept in the database.

The rest of this section is divided into 3 main parts. Subsystem Decomposition, Hardware and Software Mapping, and Design Details. In Subsystem Decomposition, the general view of the system will be given and how each subsystem interacts with each other will be explained. In Hardware and Software Mapping, how the system functions in terms of software and hardware will be displayed. Finally, in Design Details, detailed information on how each subsystem works will be explained. Details on tools, libraries, frameworks, etc. will be given in the Development/Implementation Details section of the report.

3.2 Subsystem Decomposition

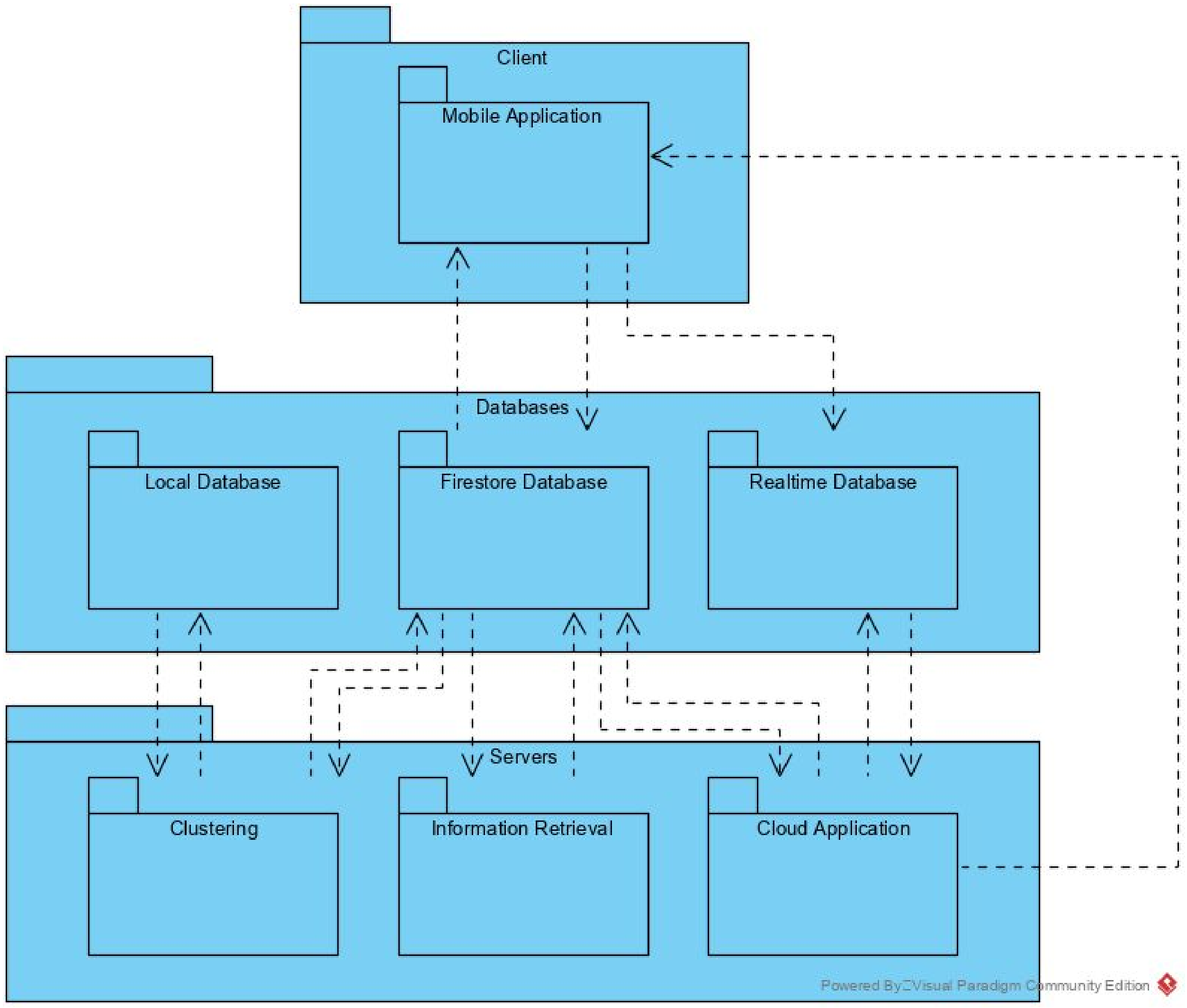


Figure 1 - Subsystem Decomposition Diagram

In the highest perspective, there are three server-side subsystems which are Clustering, Information Retrieval and Cloud Application, and one client-side subsystem which is Mobile Application. These subsystems are connected to each other through some central databases and they are truly modular and independent of each other. The Mobile Application subsystem is the only subsystem that the user has any interactions with. The Information Retrieval and the Clustering subsystems work in the background in our servers. The Information Retrieval subsystem is responsible for collecting and storing the raw news data for later analysis. The Clustering subsystem is responsible for processing the raw news data and grouping the related news by using batch and incremental clustering techniques coherently. Finally, the Mobile Application subsystem is responsible for presenting the news and our analysis of the news to the user.

3.3 Design Details

Front-end Application Subsystem

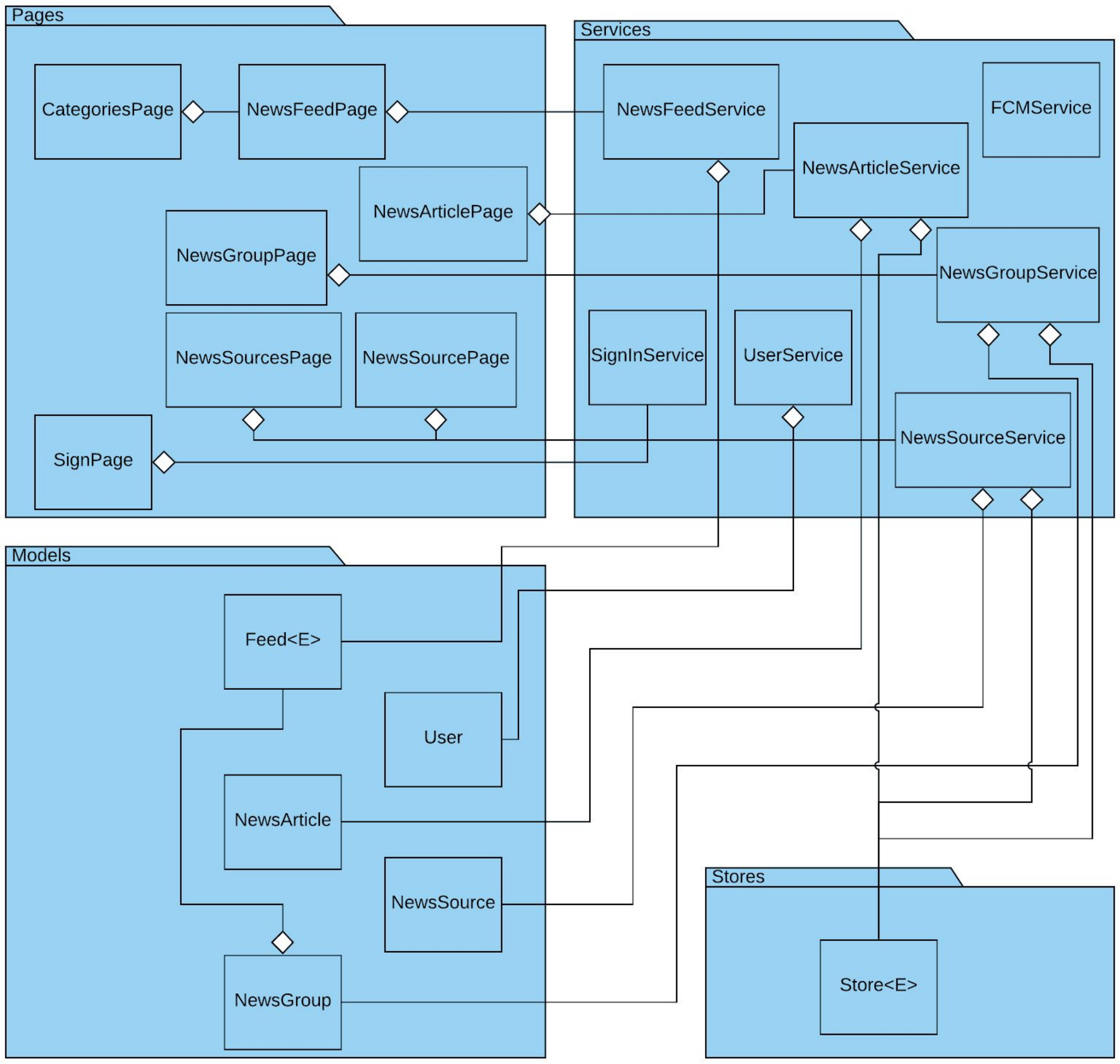


Figure 2 - Class Diagram of the Front-end Application Subsystem

Figure 2 shows the overall dependencies and connections between classes and packages. Figures 3 to 6, show the interfaces of the classes, their attributes, constructors, and methods. As seen in figure 2, each package fulfills a different role in the codebase. The pages package contains the screens that are shown inside the application. Each class represents one page or screen. The services package contains the functionality of our application. All business logic resides in this package. Stores package contains the Store class which is a custom data structure design to hold our models. Finally, the Models package, contains all models in our application. These models represent our data, from users to news stories to newsgroups. Models do not have inherent functionally and they only have helper functions that make accessing or updating models data easier.

Models are kept in stores in order to guarantee that no duplicate model can ever exist in the lifecycle of the application. Models are kept in a hashtable inside the stores for quick access. The Store is a general data structure that can keep any of our models. Stores are created and maintained by services. Only services have direct access to the stores and other classes can only access stores via services. Services handle all business logic in our application. The services are created to handle the functionalities that are associated with our models. In other words, services give functionality to our models. Pages are the only way for our users to interact with our application. Pages display, add, update, etc. models using services. By using this design approach we can ensure that there is no duplicate data or model between pages and every model that is displayed in any of our pages is the same model for all pages. Additionally, because each model is unique we can easily synchronize our application models with entities in our databases.

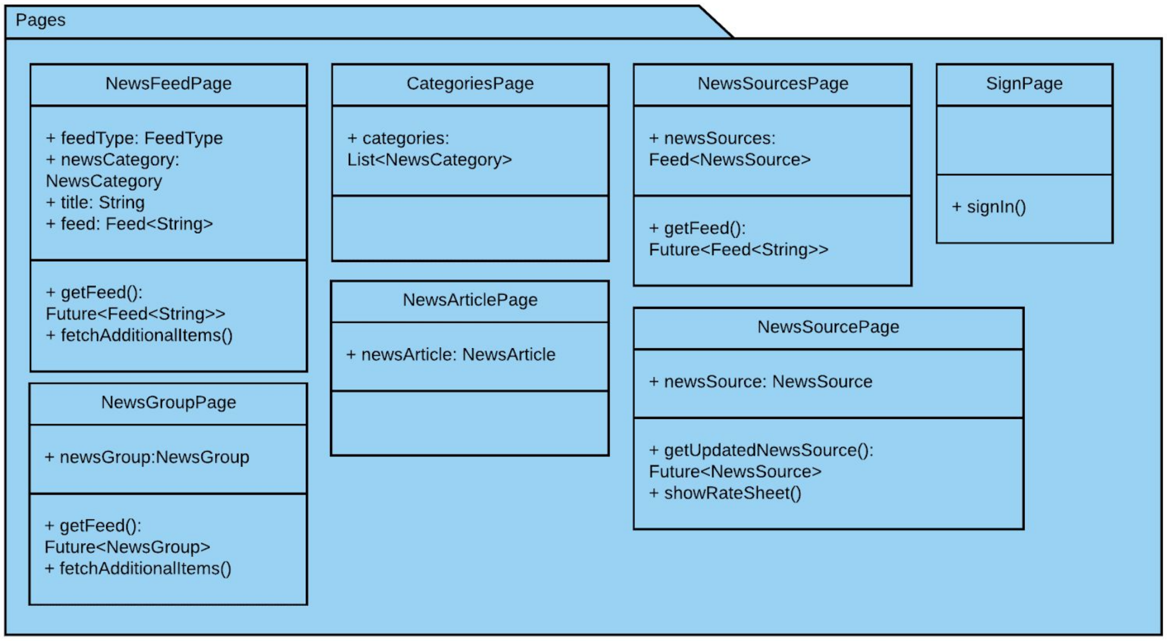


Figure 3 - Class Diagram of Pages Package

- NewsFeedPage is a generic page that is used to display any feed of newsgroups.
- NewsGroupPage shows news articles in a timeline.
- NewsArticlePage shows detailed information on a selected news story.
- Categories Page display our categories so that user can choose and display newsgroups on a selected category.
- NewsSourcePage shows detailed information about a selected news source.
- NewsSourcesPage displays our news sources and shows detailed information about our news sources.
- SignPage is where users sign in to our application.

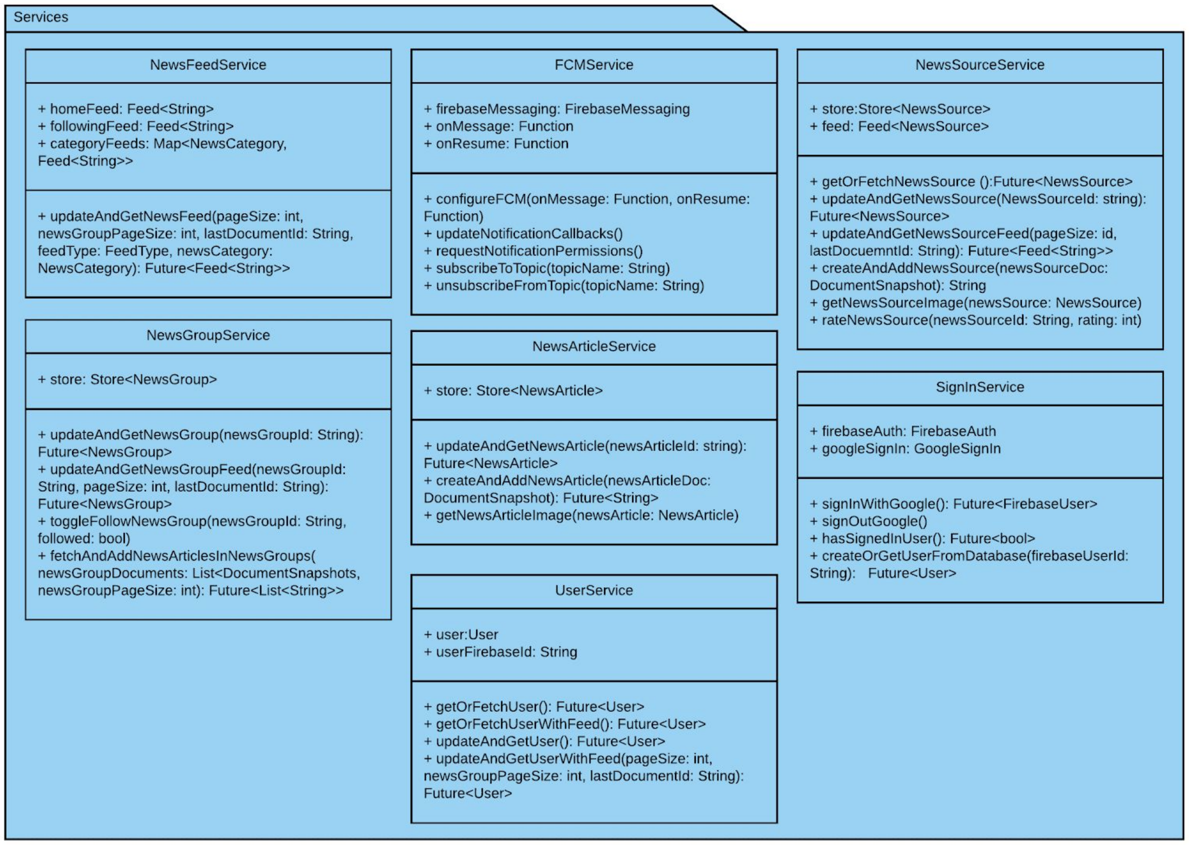

Services
NewsFeedService
+ homeFeed: Feed<String>
+ followingFeed: Feed<String>
+ categoryFeeds: Map<NewsCategory, Feed<String>>
+ updateAndGetNewsFeed(pageSize: int, newsGroupPageSize: int, lastDocumentId: String, feedType: FeedType, newsCategory: NewsCategory): Future<Feed<String>>
FCMService
+ firebaseMessaging: FirebaseMessaging
+ onMessage: Function
+ onResume: Function
+ configureFCM(onMessage: Function, onResume: Function)
+ updateNotificationCallbacks()
+ requestNotificationPermissions()
+ subscribeToTopic(topicName: String)
+ unsubscribeFromTopic(topicName: String)
NewsSourceService
+ store:Store<NewsSource>
+ feed: Feed<NewsSource>
+ getOrFetchNewsSource ():Future<NewsSource>
+ updateAndGetNewsSource(NewsSourceId: string): Future<NewsSource>
+ updateAndGetNewsSourceFeed(pageSize: id, lastDocuemntId: String): Future<Feed<String>>
+ createAndAddNewsSource(newsSourceDoc: DocumentSnapshot): String
+ getNewsSourceImage(newsSource: NewsSource)
+ rateNewsSource(newsSourceId: String, rating: int)
NewsGroupService
+ store: Store<NewsGroup>
+ updateAndGetNewsGroup(newsGroupId: String): Future<NewsGroup>
+ updateAndGetNewsGroupFeed(newsGroupId: String, pageSize: int, lastDocumentId: String): Future<NewsGroup>
+ toggleFollowNewsGroup(newsGroupId: String, followed: bool)
+ fetchAndAddNewsArticlesInNewsGroups(newsGroupDocuments: List<DocumentSnapshots, newsGroupPageSize: int): Future<List<String>>
NewsArticleService
+ store: Store<NewsArticle>
+ updateAndGetNewsArticle(newsArticleId: string): Future<NewsArticle>
+ createAndAddNewsArticle(newsArticleDoc: DocumentSnapshot): Future<String>
+ getNewsArticleImage(newsArticle: NewsArticle)
SignInService
+ firebaseAuth: FirebaseAuth
+ googleSignIn: GoogleSignIn
+ signInWithGoogle(): Future<FirebaseUser>
+ signOutGoogle()
+ hasSignedInUser(): Future<bool>
+ createOrGetUserFromDatabase(firebaseUserId: String): Future<User>
UserService
+ user:User
+ userFirebaseId: String
+ getOrFetchUser(): Future<User>
+ getOrFetchUserWithFeed(): Future<User>
+ updateAndGetUser(): Future<User>
+ updateAndGetUserWithFeed(pageSize: int, newsGroupPageSize: int, lastDocumentId: String): Future<User>

Figure 4 - Class Diagram of Services Package

- NewsFeedService provides all functionality that involves newsgroup feeds.
- NewsGroupService provides all functionality that involves newsgroups.
- NewsArticleService provides all functionality that involves news articles.
- NewsSourceService provides all functionality that involves news sources.
- UserService provides all functionality that involves our users.
- FCMService provides all functionality that involves notifications.
- SignInService provides all functionality that is needed for sign-in and authentication.

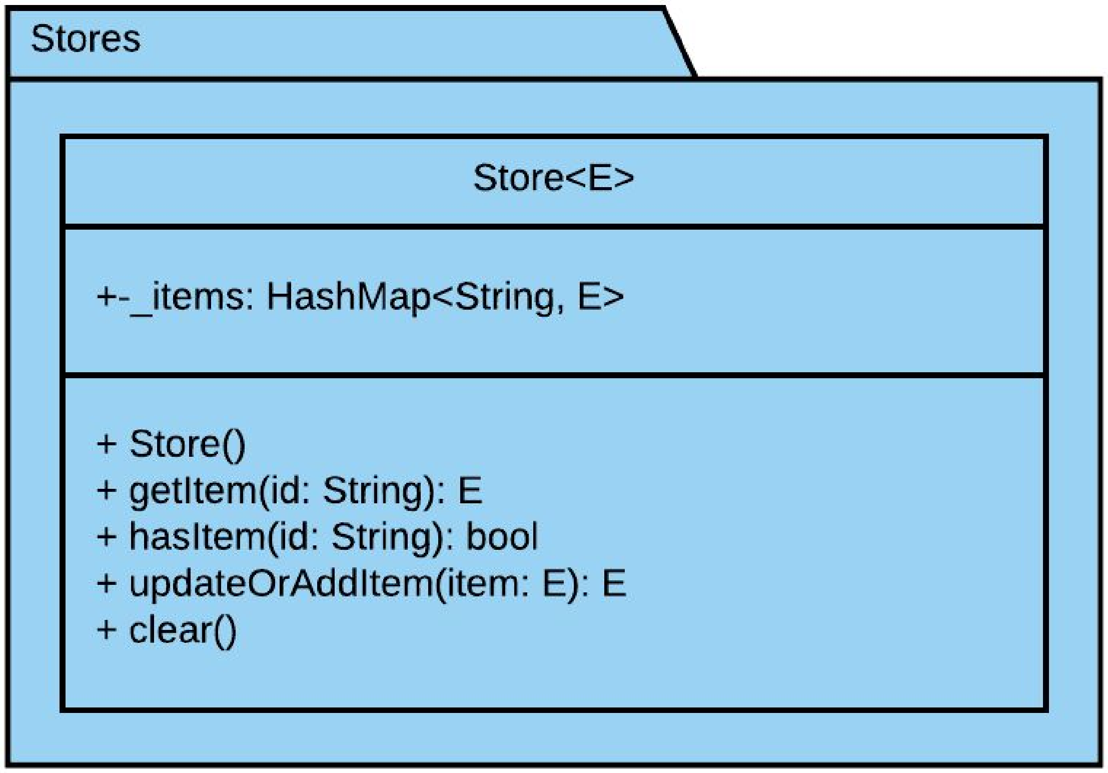

Stores
Store<E>
+-_items: HashMap<String, E>
+ Store()
+ getItem(id: String): E
+ hasItem(id: String): bool
+ updateOrAddItem(item: E): E
+ clear()

Figure 5 - Class Diagram of StroesPackage

- Store class is a custom data structure to handle the storage of models in memory. Stores guarantee that no duplicate models evert exist in our application. Additionally, stores provide fast and easy access to any model for any service.

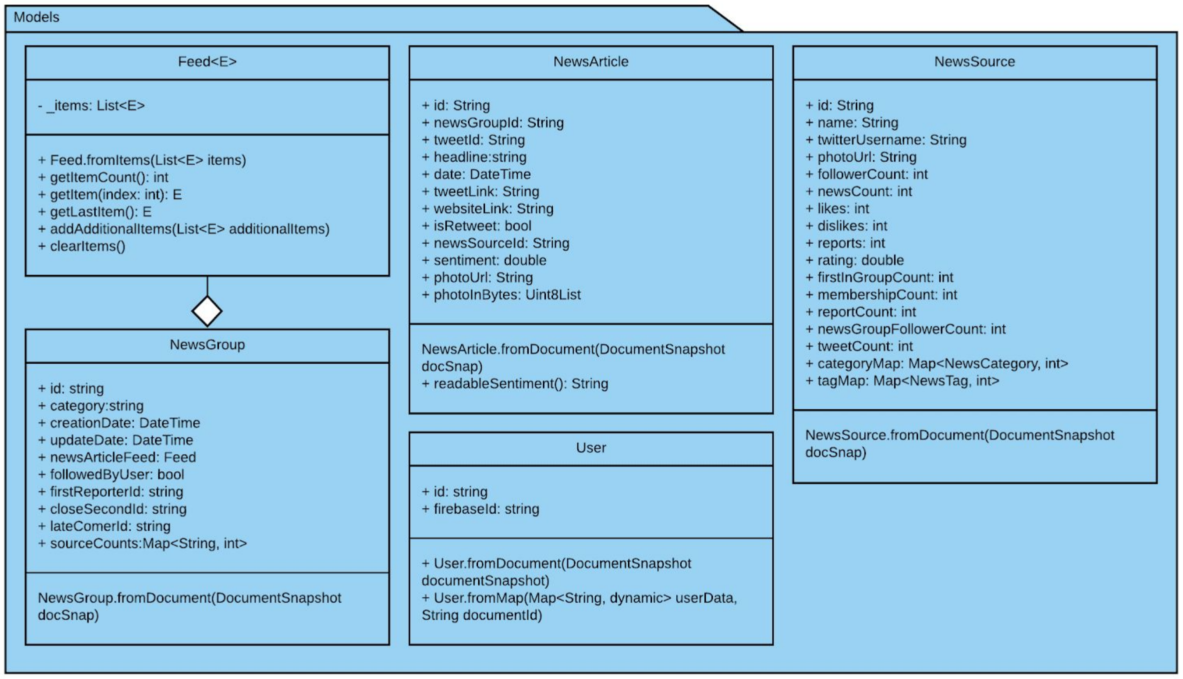


Figure 6 - Class Diagram of Models Package

- The Feed is a general class that helps model feeds in modern applications. Feeds act like improved lists and provide functionalities for services to access their information easier and faster. Additionally, using the feed class allows most of our services and pages to reuse code.
- The NewsGroup class represents our newsgroups. The NewsGroup class has no methods and can be constructed from a database document.
- The NewsArticle class represents our news articles. The NewsArticle class has only one helper method to convert its numeric sentiment value into a readable string. The NewsArticle class can be constructed from a database document.
- The NewsSource class represents our newsgroups. The NewsSource class has no methods and can be constructed from a database document.
- User class represents our users. The User class has no methods and can be constructed either from a database document or from a map.

Clustering Subsystem

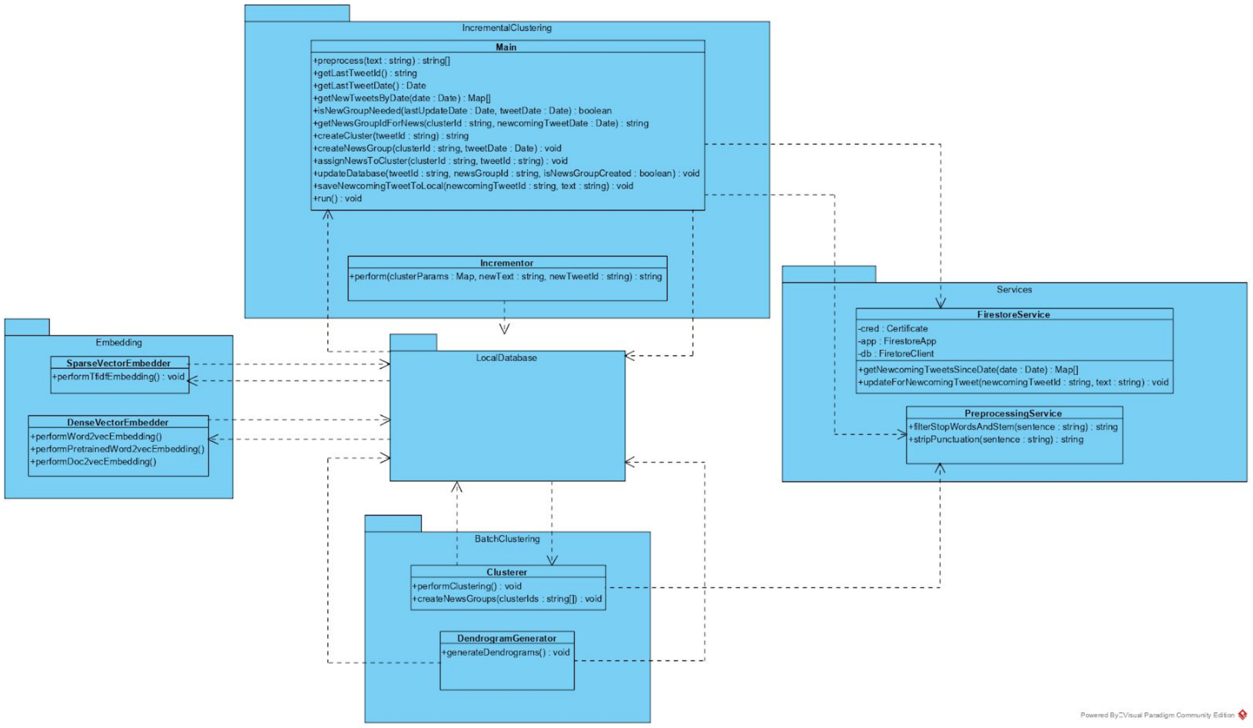

IncrementalClustering
Main
+preprocess(text : string) : string[]
+getLastTweetId() : string
+getLastTweetDate() : Date
+getNewTweetsByDate(date : Date) : Map[]
+isNewGroupNeeded(lastUpdateDate : Date, tweetDate : Date) : boolean
+getNewsGroupIdForNews(clusterId : string, newcomingTweetDate : Date) : string
+createCluster(tweetId : string) : string
+createNewsGroup(clusterId : string, tweetDate : Date) : void
+assignNewsToCluster(clusterId : string, tweetId : string) : void
+updateDatabase(tweetId : string, newsGroupId : string, isNewsGroupCreated : boolean) : void
+saveNewcomingTweetToLocal(newcomingTweetId : string, text : string) : void
+run() : void
Incrementor
+perform(clusterParams : Map, newText : string, newTweetId : string) : string
Services
FirestoreService
-cred : Certificate
-app : FirestoreApp
-db : FiretoreClient
+getNewcomingTweetsSinceDate(date : Date) : Map[]
+updateForNewcomingTweet(newcomingTweetId : string, text : string) : void
PreprocessingService
+filterStopWordsAndStem(sentence : string) : string
+stripPunctuation(sentence : string) : string
Embedding
SparseVectorEmbedder
+performTfidfEmbedding() : void
DenseVectorEmbedder
+performWord2vecEmbedding()
+performPretrainedWord2vecEmbedding()
+performDoc2vecEmbedding()
LocalDatabase
BatchClustering
Clusterer
+performClustering() : void
+createNewsGroups(clusterIds : string[]) : void
DendrogramGenerator
+generateDendrograms() : void
Powered By Visual Paradigm Community Edition

Figure 7 - Class Diagram of the Clustering Subsystem

The main feature of the architecture of this subsystem is that most of the classes are runnable script files and communicate over the file system that is the Local Database. The reason this subsystem wasn't implemented as a single program is that the classes of the subsystem don't require to work in an order and most of them should run when it is necessary. For example, the performClustering() method in the Clusterer class in the BatchClustering package is for performing batch clustering and creating the initial clusters. To be able to do that, it needs the embedded document vectors to compare. However, it can't just call the methods in the Embedding package because a human eye must analyze the dendrograms generated by the generateDendrograms() method in the DendrogramGenerator class in the BatchClustering package. Therefore, the script files of each of these classes run asynchronously and optionally.

This subsystem adapts other subsystems by calling the run() method in the Main class frequently. This method checks the database for new-coming news and processes them if found. The process is divided into three methods: preprocessing, embedding, and clustering.

Since the similarities between the texts are computed, the texts that are compared should follow a standard. In order to obtain the standard, the following techniques are applied for preprocessing:

- Removing stop-words,
- Word stemming,
- Removing punctuations

Because stop-words, non-stemmed words, and punctuation marks can cause deceiving information. These techniques are applied using the freestanding methods in the Preprocessing class which is in the Services package.

In the final version of the project, we used the performTfidfEmbedding() in the SparseVectorsEmbedder class in the Embedding package since tf-idf embedding method is our final decision for this part of the project. The steps that we followed to obtain this decision are given in the Testing Details section. The other methods in the Embedding package are to perform other embedding methods such as Word2vec and Doc2vec which are the comparison elements for the testing. The details about these embeddings are given in the Development/Implementation Details section.

Since breaking news always keeps coming and the system streams them in real-time, incremental clustering, that uses online learning, is the key point for this system. However, it is not a good idea to apply only incremental clustering to the breaking news because their lifetime is 48 hours at maximum, just like the clusters they belong to. This means that the breaking news leads to concept drift very frequently and it would be wrong to search a cluster among the ones that were created several weeks ago for a new-coming one. In this case, applying batch clustering on the data that belong to the same concept can handle this drift problem and can have the news with the same concept grouped in the same clusters. In order to apply batch clustering the performClustering() method in the Clusterer class in the BatchClustering package is called when it is necessary. Yet, there is a problem with batch clustering as well. When the batch clustering is applied, all the current clusters get refreshed and all the dynamics of the system changes. Most importantly, there would be no static clusters to present to the user which could reduce the user interaction if the batch clustering was applied very frequently. Therefore, we tried to find a balance between the batch clustering and the incremental clustering and developed the Newsgroup method.

The Newsgroup method is a multi-layer clustering process which combines the batch clustering and the incremental clustering. In this method, there are newsgroups that represent a unique cluster and the users are able to see these newsgroups instead of the clusters. The newsgroups are static and a member of a newsgroup can't be removed or transferred to another newsgroup in any case. In order to adapt the concept drifts, the lifetime of a newsgroup is a fixed time which is 48 hours for this project according to the observations made on the concept drift frequency of the breaking news. After this time passes since the newsgroup was generated, the newsgroup shuts down and a new newsgroup for the main cluster is generated when a new headline is assigned to that cluster. This method allows the system to perform the incremental clustering and the batch clustering coherently. Even the batch clustering recasts the existing clusters with the new ones, the newsgroups remain the same and the user can keep watching them. So, performing batch clustering in order to avoid concept drifts doesn't affect the stability of the system. Besides that, the system can also continue online learning. When a new headline is streamed, the incremental clustering handles that data using the existing cluster and newsgroups by following the procedure that will be explained in the Development/Implementation Details section in detail.

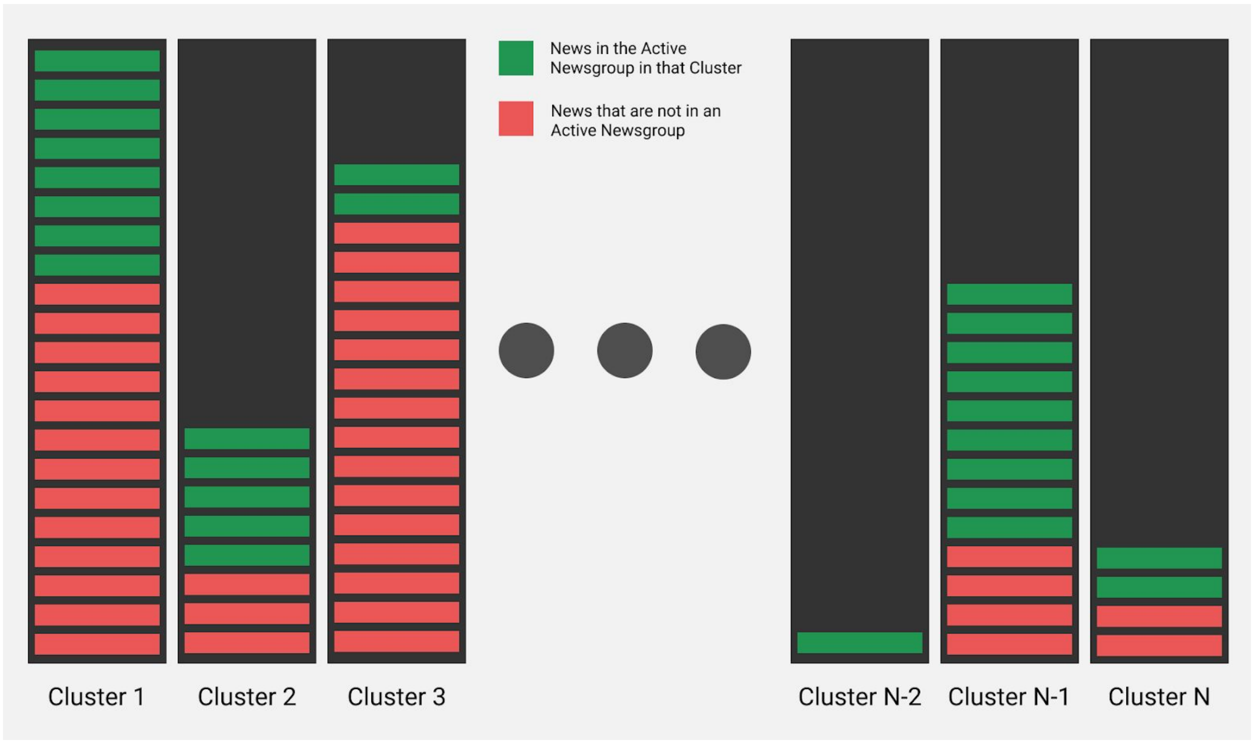

News in the Active Newsgroup in that Cluster
News that are not in an Active Newsgroup
Cluster 1
Cluster 2
Cluster 3
Cluster N-2
Cluster N-1
Cluster N

Figure 8 - A sample representation of News Group method

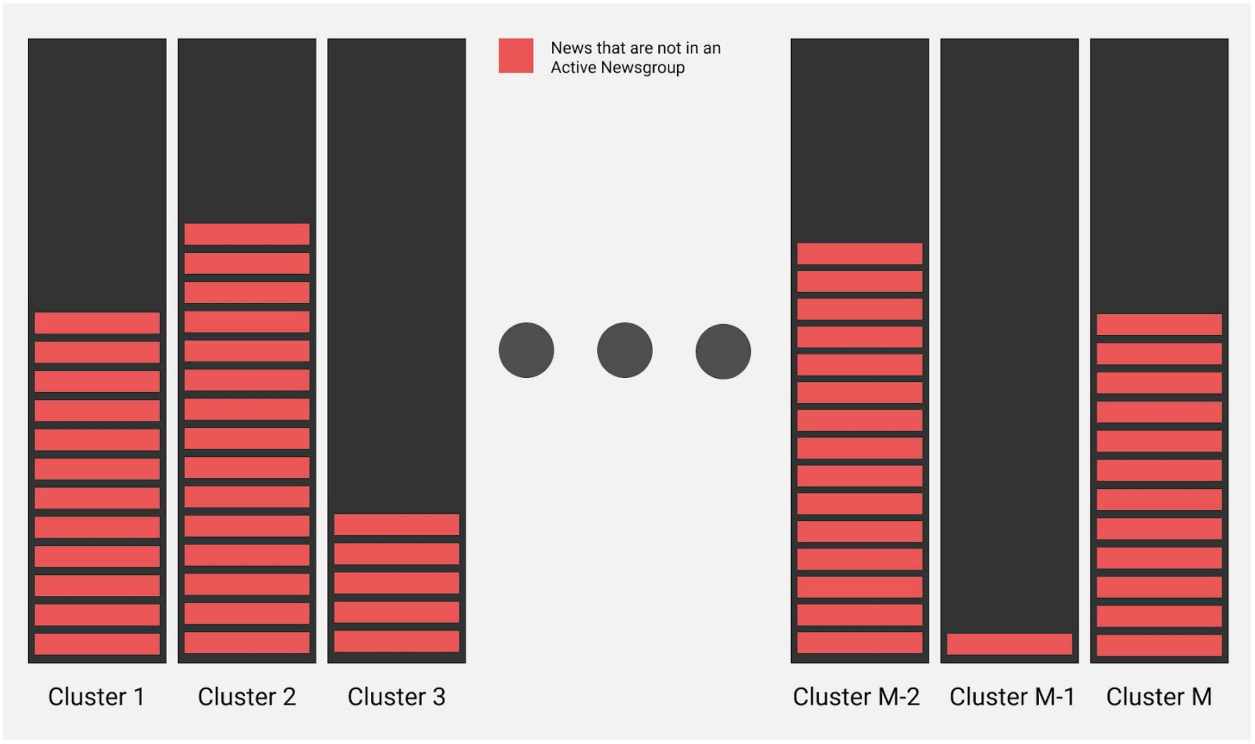


Figure 9 - A sample representation of News Group method after batch clustering is applied onto the state in Figure 8

Information Retrieval Subsystem

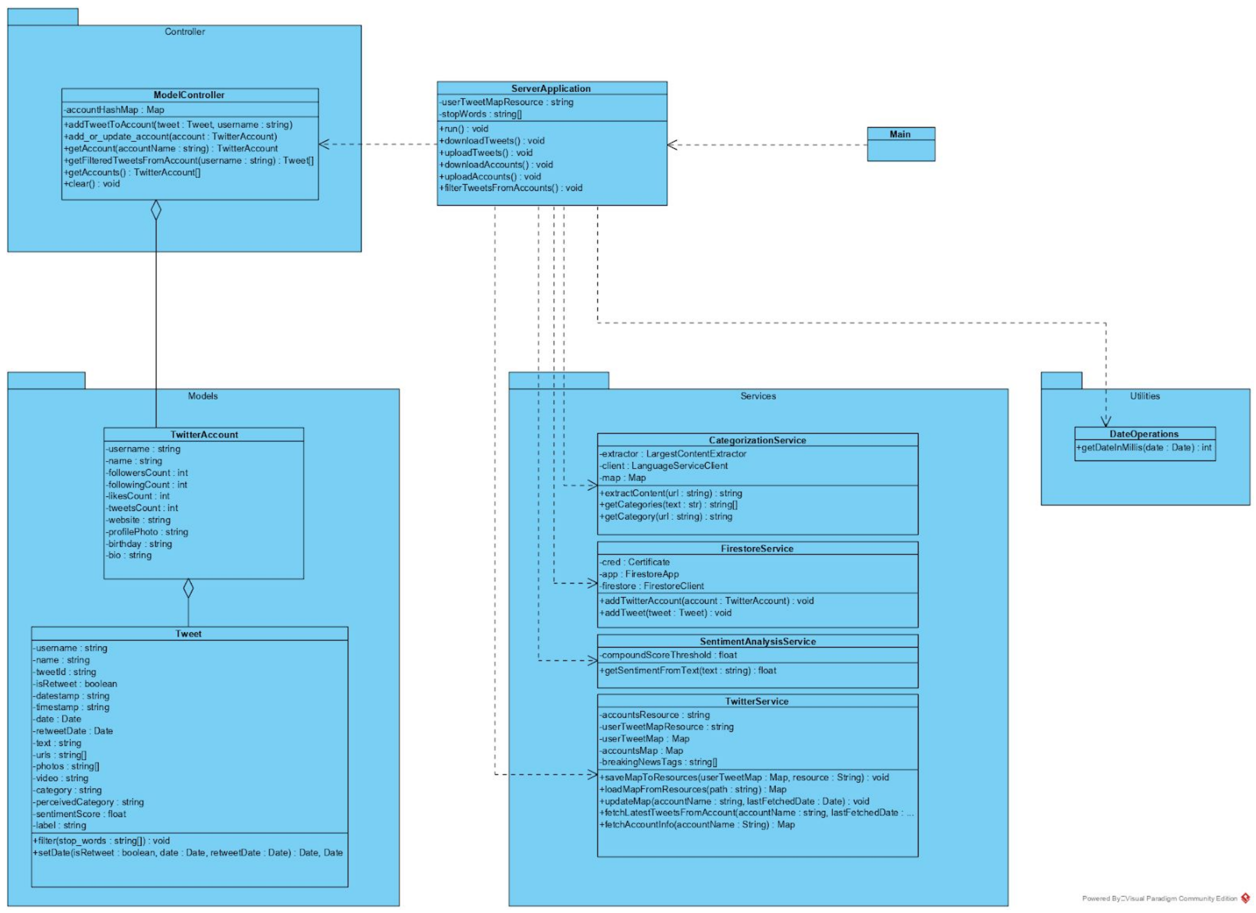

Controller
ModelController
-accountHashMap : Map
+addTweetToAccount(tweet : Tweet, username : string)
+add_or_update_account(account : TwitterAccount)
+getAccount(accountName : string) : TwitterAccount
+getFilteredTweetsFromAccount(username : string) : Tweet[]
+getAccounts() : TwitterAccount[]
+clear() : void
ServerApplication
-userTweetMapResource : string
-stopWords : string[]
+run() : void
+downloadTweets() : void
+uploadTweets() : void
+downloadAccounts() : void
+uploadAccounts() : void
+filterTweetsFromAccounts() : void
Main
Models
TwitterAccount
-username : string
-name : string
-followersCount : int
-followingCount : int
-likesCount : int
-tweetsCount : int
-website : string
-profilePhoto : string
-birthday : string
-bio : string
Tweet
-username : string
-name : string
-tweetId : string
-isRetweet : boolean
-datestamp : string
-timestamp : string
-date : Date
-retweetDate : Date
-text : string
-urls : string[]
-photos : string[]
-video : string
-category : string
-perceivedCategory : string
-sentimentScore : float
-label : string
+filter(stop_words : string[]) : void
+setDate(isRetweet : boolean, date : Date, retweetDate : Date) : Date, Date
Services
CategorizationService
-extractor : LargestContentExtractor
-client : LanguageServiceClient
-map : Map
+extractContent(url : string) : string
+getCategories(text : str) : string[]
+getCategory(url : string) : string
FirestoreService
-cred : Certificate
-app : FirestoreApp
-firestore : FirestoreClient
+addTwitterAccount(account : TwitterAccount) : void
+addTweet(tweet : Tweet) : void
SentimentAnalysisService
-compoundScoreThreshold : float
+getSentimentFromText(text : string) : float
TwitterService
-accountsResource : string
-userTweetMapResource : string
-userTweetMap : Map
-accountsMap : Map
-breakingNewsTags : string[]
+saveMapToResources(userTweetMap : Map, resource : String) : void
+loadMapFromResources(path : string) : Map
+updateMap(accountName : string, lastFetchedDate : Date) : void
+fetchLatestTweetsFromAccount(accountName : string, lastFetchedDate : ...
+fetchAccountInfo(accountName : String) : Map
Utilities
DateOperations
+getDateInMillis(date : Date) : int
Powered By Visual Paradigm Community Edition

Figure 10 - Class Diagram of the Information Retrieval Subsystem

The Information Retrieval Subsystem is a server application which is responsible for fetching, processing and uploading the latest news published on Twitter to our database in every minute. In order to fetch, process and upload these breaking news from relevant sources we use Services. The Services package includes several services which are dedicated for different purposes. These are Categorization Service, Firestore Service, Sentiment Analysis Service and Twitter Service.

Each service is powered by an outside tool and is accessed in our Server Application Class where the main application flow is implemented. To handle the information flowing inside the application, we have a Model Controller which basically controls our instances of two model classes, Twitter Account and Tweet. First, Twitter Service fetches the latest account information along with their latest news published and forms our Twitter Account and Tweet models for each news source. Then, with the help of Model Controller, the contents of each Tweet is filtered from stop words so that they are clean and ready to use. Later on, with the help of Categorization and Sentiment Analysis Services, a relevant category and sentiment is assigned to our each Tweet model. Finally, the processed models and the account information is uploaded to the Firestore database by the Firestore Service. The process mentioned is repeated in each minute in our main class, which is only responsible for initializing our server application with proper resources and calling the "Run" method of it.

- TwitterAccount is the model class for an individual news source's Twitter Account
- Tweet is the model class for an individual news published on Twitter
- CategorizationService is the class responsible for applying categorization to our tweet contents
- SentimentAnalysisService is the class responsible for applying sentiment analysis to our tweet contents
- FirestoreService is the class responsible for uploading the processed tweets to our database
- TwitterService is the class responsible for fetching the latest information from Twitter
- DateOperations is the class responsible for converting dates to desired formats

4. Development/Implementation Details

4.1 Information Retrieval Subsystem

The Information Retrieval Subsystem is implemented in Python Programming Language. The application fetches, processes, and sends the data with the help of several libraries. The Python packages that are installed are FirebaseAdmin, Twint, Google Cloud Language, Vader Sentiment, and BoilerPipe.

- **Firebase Admin**

  This library is used for accessing both Google's FireStore and Realtime Databases. It is one of the core libraries that we need to use, as the upload of the data being fetched to the database is the most important function of this subsystem that leads other subsystems to work. [8]

- **Twint: An Advanced Twitter Scraping Tool**

  Due to our unsuccessful attempts for accessing the Twitter Developer API we have found several alternative web scrapers and the best of them was Twint. "Twint is an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter's API." Although the library is not stable, we are able to get detailed information about any tweet that is published by an account and upload it to our database after the process of stop word removal, category extraction, and sentiment analysis.[9]

- **Google Cloud Language**

  Google provides a variety of cloud tools to its users. In our application, we need to display the categories of the news articles and we have found a useful Google Cloud library Google Cloud Language which is able to extract the category of a given text. With the help of this library, we are able to extract the content of news, if and only if they are linked to a specific article, as a word limit count of 20 can mostly not be reached by using only the headlines of the news.[10]

- **VADER Sentiment**

  "VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media." With the help of this tool we are able to provide an estimate of how positive or negative a news article content is. However, until we are not certain about the results, we are not planning to integrate this tool in the release version.[11]

- **Boiler Pipe**

  Boiler Pipe is a python library that fetches the main content of an HTML Web Page. We use this library in order to extract the news article from the website so that we can provide it as an input to our categorization and sentiment analysis functions.[12]

Additionally, the information to be fetched from Twitter is being decided by a variety of JSON resource files. For example, a JSON file is used for saving the last time news was fetched from a breaking news account so that the next fetching process could continue with getting only the latest news published. Another JSON file is used for identifying the characteristics of the sources on whether they are specific breaking news sources or they require detailed filtering for retrieving only the breaking news from the whole content.

4.2 Google Firebase Cloud Application

The cloud application is implemented in the TypeScript programming language. It involves different types of triggers that listen to the database actions and perform necessary calculations to update related documents. These triggers run when OnUpdate or OnCreate actions are taken on the specific Firestore documents such as News, News Source, User Votes, and User Reports. Additionally, to these triggers we use Schedule Functions to perform actions on the database in a periodic manner. We use them to close the newsgroups and collect their overall information. Finally, our cloud application forms a bridge between Google's Firestore and Realtime Databases by using transactions to update countable information. The triggers perform necessary database transactions on Realtime Database to store the updated count of an operation or a specific type of information. To note, cloud functions are implemented in an idempotent fashion so that a trigger can not affect the documents multiple times. This is due to Google's policy of sending a trigger request at least one time to make sure the information is delivered. By keeping the latest trigger event's id, we block the current function to a duplicate event having the same event id.

4.3 Front-end Applications

Front-end applications were developed using the Flutter framework and the Dart programming language. Flutter framework is Google's cross-platform native app development solution for both mobile and web. With Flutter, we only need to develop and maintain a single codebase for every platform which makes the development extremely simple and maintainable. Dart is also developed and maintained by Google. Dart is somewhat a mix between Java and Javascript and could be considered as the best of both worlds for our use case. Dart also has a package library that is maintained by both the community and Google. These packages handle most of the common use cases for mobile applications.

Firebase Packages

These packages allow our front-end Flutter application to communicate with our Firebase back-end. There are various packages for each function that is provided by Firebase. Each package is separate in order to keep the application size to the minimum. Here are the packages that we have used.

Firebase Database

This package is used to communicate with the realtime database in our Firebase backend. The realtime database is used to store the ratings of news sources. [13]

Cloud Firestore

This package is used to communicate with the firestore database in our Firebase backend. The firestore database stores almost every document in our system from news stories to news sources to users. [14]

Firebase Auth

This package is used to create an authentication system for our application. As we need to keep track of users and also keep our user information secure, we decided to use the authentication system provided by the Firebase backend. This package allows us to synchronize the state of the account in the back-end with our front-end. Additionally, with this authentication system, we can create server-side security rules to ensure that no one can access another user's information. [15]

Google Sign in

This package is used in combination with the Firebase Auth package to allow users to sign-up to our application using their google accounts with one click. As signing-up to various different applications is an obnoxious process we wanted to provide our users with the most simple solution as possible. [16]

Firebase Messaging

This package is used to receive notifications from Firebase. We notify our users when a news story is added to a newsgroup that they are following. To send a notification to the correct users is very important. There are 3 possible events in the notification process. The “onLaunch”, “onResume”, and “onMessage” event. Each event needs a different way to be handled properly. This package provides everything in order to program every step of the notification process. [17]

Flushbar

This package is used in combination with the Firebase Messaging package. When a notification is received while the application is open, we need a way of showing the notification in a form that the user is familiar with. Flushbar package provides widgets that enable us to show notifications in our application that are similar to popular applications such as WhatsApp and Instagram. [18]

Google Fonts

This package provides our application with all the fonts that we use. We have access to all the free fonts in the Google Fonts website [19]. This package helped us try out different styles for our application with speed and ease. [20]

Webview Flutter

This package is used to show the websites of news stories, news sources, tweets, etc. inside our application. Our users can visit these websites without opening their browsers, or changing applications. [21]

Eva Icons

This package provides our application with the Eva Icons [22]. Most of the icons used in our user interface are from this package. [23]

4.4 Clustering Subsystem

Embedding

First of all, there are two types of embedding to choose from. One is word-based embedding and the other one is document-based embedding. Word-based embedding is an embedding method that computes a feature vector for each word in the corpus. Document-based embedding is another embedding method that computes a feature vector for each document in the corpus. To be able to compare the results with respect to the models with varying complexities, embedding methods that construct dense vectors and sparse vectors were used.

In order to compute the sparse vectors, we used tf-idf with a smoothing method which stands for *term frequency-inverse document frequency*. In order to get the tf-idf weights, first, we computed the following two functions using smoothing:

$$tf(t, d) = 1 + log(N_t^{\,d}) \qquad \text{(Eq. 1)}$$

where $N_t^{\,d}$ is the number of times the term occurs in document d and

$$idf(t) = log(\tfrac{1+M}{1+M_t}) \qquad \text{(Eq. 2)}$$

where M is the number of all documents and M_t is the number of documents that contain the term t.

Here Eq. 1 indicates the term frequency of a term in a document which is divided by the length of the document as a way of normalization. Eq. 2 indicates the importance of the term in the corpus with respect to the ratio of the number of all documents to the number of documents that contain that term.

At last, we got the results of $tf(t, d) \times idf(t)$ for all $t \; and \; d$ and the matching of these results with respect to $t \; and \; d$ composed the term-document matrix that will be used to obtain the similarities between the documents [24].

In order to compute the dense vectors for word embedding, we used the word2vec method [25]. The main idea of word2vec is that two words have similar meanings if they also have a similar context. Due to these artificial meanings, word2vec constructs a vector space of several hundred dimensions using a large corpus. Since this method produces the word vectors with respect to the similarities between these words, and we use the similarities between the documents for clustering, this method seems proper for our purpose. We used a pre-trained word2vec model [26] that contains the raw vectors of 3,000,000 unique words which are obtained from Google News that has similar context with this project, and we used the streamed data which contains approximately 20,000 unique words (not including the stop-words) which is much less than the pre-trained model but contains the daily breaking news that have much less noise.

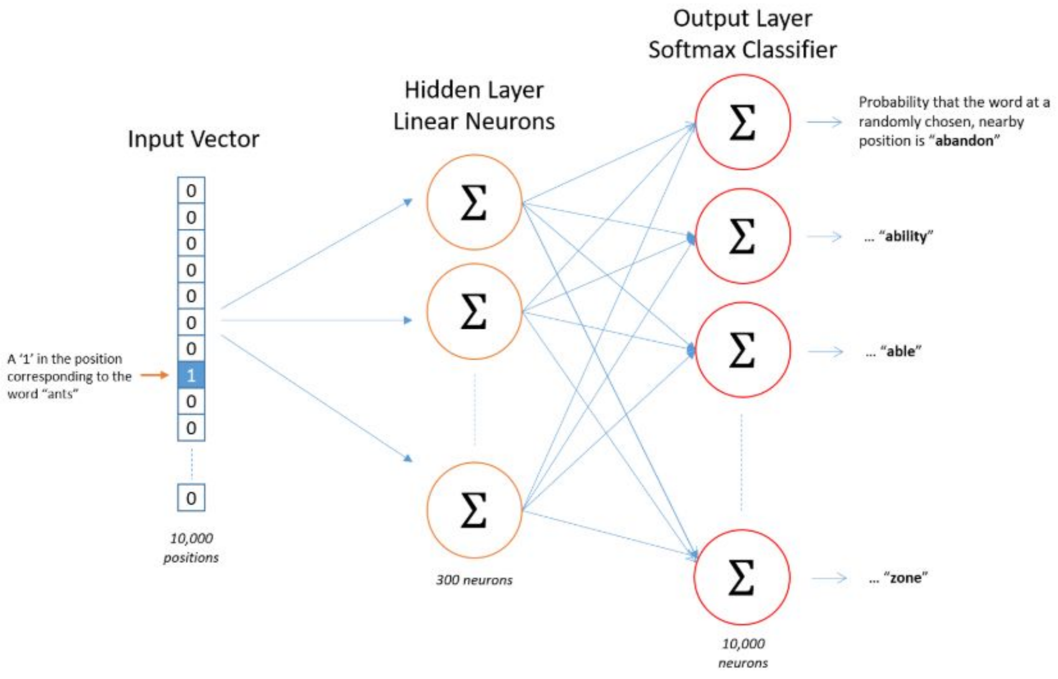


Figure 11 - A sample model for word2vec [27]

Since we cluster the documents, the actual similarity we measure is the one between the documents. Therefore, it is important to try a document-based embedding as well. We used the doc2vec method which can be considered as an extension or an upgrade to the word2vec method [X]. The doc2vec method, besides the words, uses the documents that were tagged as well. In this way, the document vectors can be accessed and used afterward which is the clustering for this project.

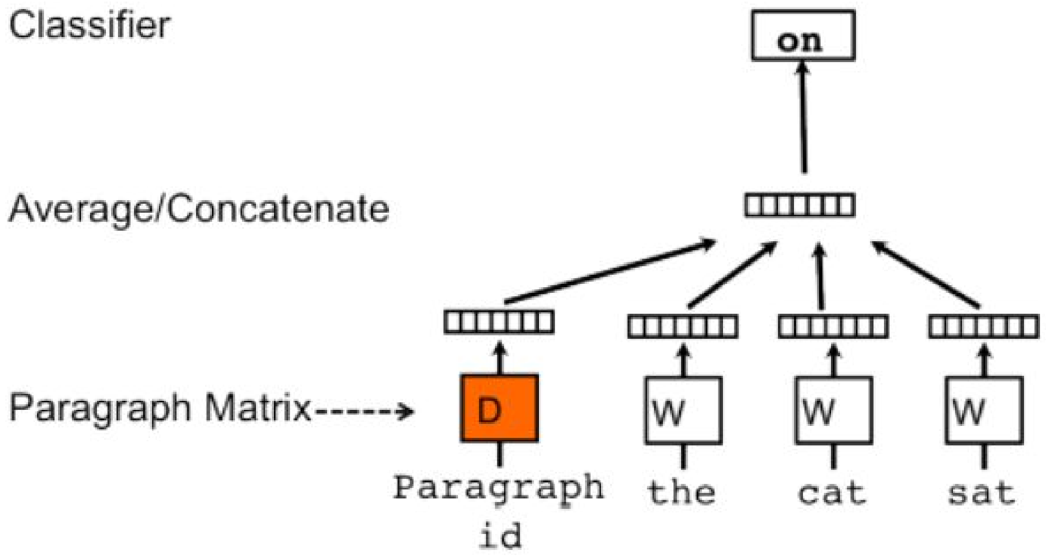


Figure 12 - A sample model for doc2vec [28]

Batch Clustering

Batch clustering is performed to pre-train the dataset before online learning starts and to avoid concept drifts when it is necessary. Since the number of clusters to be generated is uncertain and should be determined differently at each run, it is not possible to set the cluster number as a parameter to the clustering algorithm. Therefore, we decided to use a hierarchical clustering algorithm for this part of the project. To be able to determine a distance threshold that is used to separate clusters from each other by observing the dendrograms, we chose the agglomerative clustering which is a type of hierarchical clustering and where pairs of clusters get merged consequently in a bottom-up fashion [29].

Before starting the clustering process, it is needed to construct the distance matrix that defines the distances between the embedded documents. In order to measure distances, we assigned the most proper metric for the embedding method. For tf-idf and doc2vec methods, we preferred the cosine distance metric since the similarity between documents shouldn't be correlated with the length of the vectors. For the word2vec method, the cosine distance metric can't be used because the word2vec models don't store the document vectors but only word vectors. In order to handle this situation, we decided to use the word mover's distance (WMD) metric. The intuition behind WMD suggests that there is a relationship between the distances and the degree of semantic meaningfulness of the embedded words. It utilizes this relationship and treats the text documents as a weighted cloud of points of embedded words. The distance between document A and document B is calculated by summing up the minimum distances that the words from document A need to travel in order to match the exact points of words in document B [30].

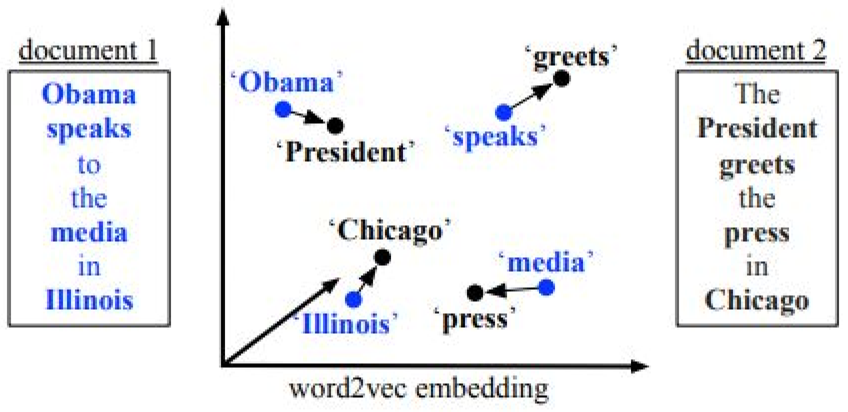


Figure 13 - A sample representation of Word Mover's Distance [30]

The agglomerative clustering algorithm does not require the number of clusters to be initially determined. However, because the number of clusters is unknown, the distance threshold and the linkage metrics need to be specified. The agglomerative clustering algorithm computes all the possible clusters one at a time and stores the possibilities in the linkage matrix and the distance threshold sets a limit for choosing which clusters should be constructed. We used the Weighted Pair Group Method with Arithmetic Mean (WPGMA) [31] linkage method for the sake of the Newsgroup method. The reason will be explained in more detail in the Incremental Clustering section.

Incremental Clustering

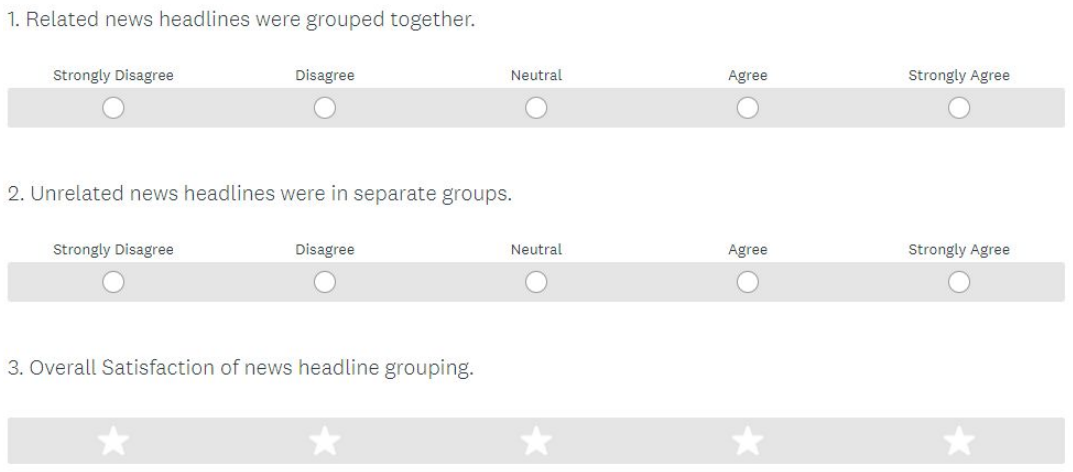
The incremental clustering is the online learning process of this system. The main purpose of this process is to take the right action when there is a new-coming news headline. There are three actions to take. The first one is to assign the headline to a cluster and to a newsgroup together. The second one is to assign it to a cluster and create a new newsgroup in that cluster. The final one is to create a new cluster and newsgroup for the headline. The deciding procedure for performing these actions is basic and straightforward. The main problem of this procedure is to decide whether the new-coming headline should be assigned to an existing cluster or to a newly created cluster. In order to decide correctly, first of all, the pairwise distance between the new-coming headline and the others should be computed. Then, the mean distances between the new-coming headline and the members of a cluster can be considered as the distance of that headline to that cluster. Computing the mean is beneficial since all of the members of a cluster are important. Considering the maximum or minimum distance between the headline and the cluster members can reduce the assigning accuracy since all of the members of a cluster are important for the assignment check and should take part in the computation. This is also the reason why we chose the WPGMA linkage method instead of the other ones such as the single or complete linkage methods [29] for batch clustering. In this way, the incremental clustering and the batch clustering manage to work in the same manner and the result that will be shown for the batch clustering will also portray the incremental clustering.

After acquiring the mean distances between the new-coming data and the clusters, we filter the ones which are less than the distance threshold used for the batch clustering in order to keep maintaining the coherency between the two clustering methods. At last, the headlines get assigned to the one with the minimum distance among the filtered clusters. If there is no filtered cluster meaning there is no cluster with the mean distance to the headline less than the distance threshold, a new cluster with a newsgroup is created and the new-coming headline gets assigned to that cluster as an initial member. If the headline is assigned to a cluster and that cluster doesn’t have an active newsgroup, then a new newsgroup is created for that headline.

5. Testing Details

Since the main approach of this project is to present the related news in the same group to users, the success of this project should be tested by measuring the accuracy of the clustering of the related news. However, we use unsupervised clustering methodology to achieve that and it is impossible to label the relatedness of the news headlines objectively. Thus, it is difficult to determine an evaluation metric for this project. The relatedness between two headlines usually depends on the human who is evaluating them. For instance, the main focus of people in 2020's first half which is the time of this project is coronavirus pandemic so about 70% of the breaking news are about this pandemic. In this context, it would be wrong to put all of the news about it to the same cluster because it would contrast with the purpose of the project since the user who uses this system couldn't follow the news efficiently. Also, there is no certain way to separate the news headlines about the coronavirus from each other. A group of people may desire to see the coronavirus news headlines clustered in the domain of countries but another group of people may want to see the coronavirus news headlines clustered in the domain of daily cases regardless of countries. Therefore, we decided to conduct a survey and evaluate the success of clusters with respect to the satisfaction of people who interact with them. Before that, we used dendrograms to observe if clustering is possible since the dendrogram illustrates the arrangement of clusters after the linkage method performed and the distance between the data points.

We conducted a survey [32] among 20 people who read the news daily to measure overall satisfaction. The thoughts of people are crucial while evaluating the performance since the main audience of this system is the educated news-readers.



1. Related news headlines were grouped together.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

2. Unrelated news headlines were in separate groups.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
○	○	○	○	○

3. Overall Satisfaction of news headline grouping.

★ ★ ★ ★ ★

Figure 14 - Survey Questions

The survey consists of three questions as shown in Figure 14. Our intention of choosing these questions is that they measure the completeness and homogeneity scores of clustering subjectively. The first question represents completeness and the second question represents homogeneity. Mathematically, the completeness and the homogeneity scores are measured using the conditional entropy between the labels of members and the cluster assignments [33]. Since it isn't possible to determine certain labels for the members as mentioned, we measured these scores using the labels that the participants approved individually. For all the scores, we calculated the weighted average of the answers of all the participants for all questions to compare the models used for clustering. For The models, we used the weighted averages obtained from the third question to measure the satisfaction of participants overall the clustering process and also to determine whether the participants care more about the completeness or homogeneity. To achieve that, we calculated the v-measure scores which are the harmonic mean of the completeness and homogeneity scores using Eq. 3. and compared them with the overall satisfaction scores. If the overall satisfaction score is between the completeness score and v-measure score, it means the participants care more about completeness. If the overall satisfaction score is between the homogeneity score and v-measure score, it means the participants care more about homogeneity.

$$v = \frac{2 \times completeness \times homogeneity}{(completeness + homogeneity)} \qquad \text{(Eq. 3) [33]}$$

For standardization, all the values of scores are over 5. To achieve that, for the first two questions, the answers "Strongly Disagree" corresponds to weight 1, "Disagree" corresponds to weight 2, "Neutral" corresponds to weight 3, "Agree" corresponds to weight 4, "Strongly Agree" corresponds to weight 5, and each star corresponds to weight 1.

5.1 Dendrograms

Dendrograms are very important for analyzing the results of linkage methods. We had the dendrograms for each model that use different embedding methods drawn. By analyzing these dendrograms, we decided which models are available for clustering and which are not. Also, we determined distance thresholds which are used for hierarchical clustering for the models that are available for clustering.

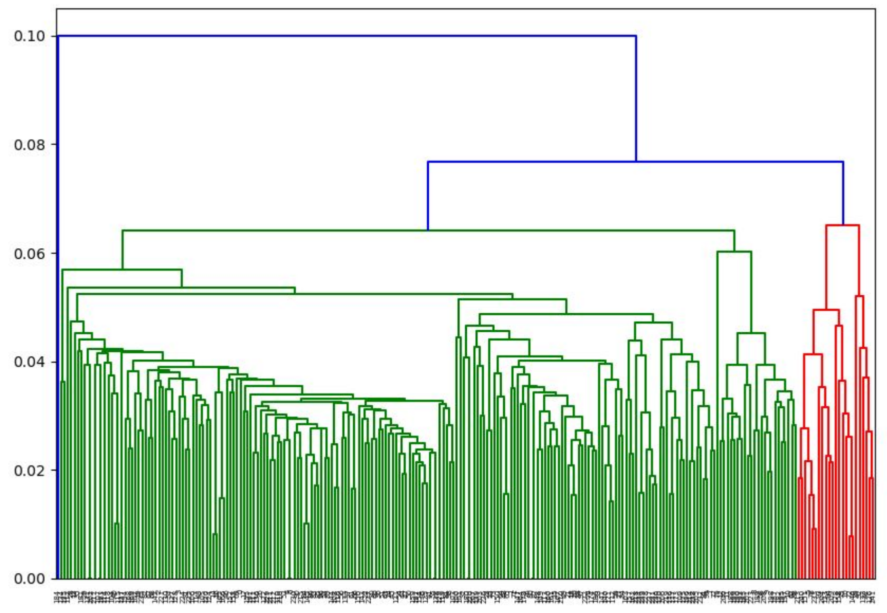


Figure 15 - Dendrogram for the weighted linkage of the word2vec model trained with streaming data

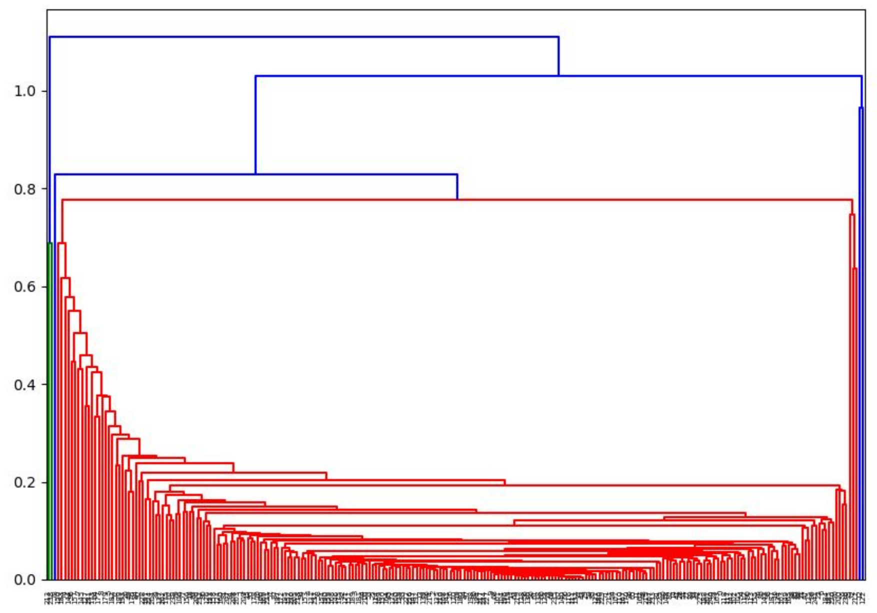


Figure 16 - Dendrogram for the weighted linkage of the doc2vec model trained with streaming data

As it can be seen in Figure 15 and Figure 16 , the cluster arrangement are very close each other with the distances that are very close to 0. The reason is that the distances between embedded vectors are also close to each other. This shows that when the doc2vec and word2vec models train with a dataset that has limited size such as the streaming data that we mined, the model most probably overfits and behaves like all the headlines are similar to each other. In this manner, we eliminated the the word2vec and doc2vec models using the streaming data since clustering won't work properly using the embedded vectors generated by these models as the dendrograms in Figure 15 and Figure 16 illustrates.

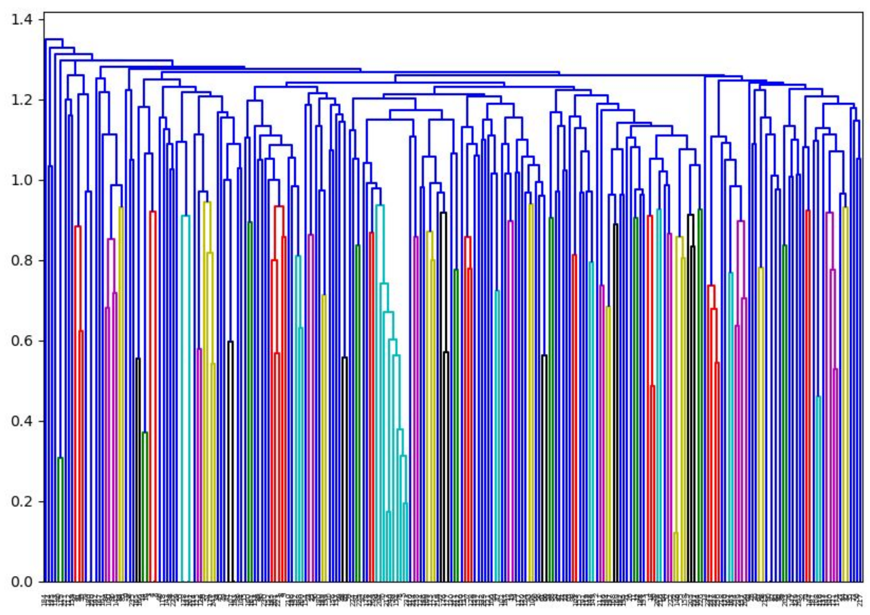


Figure 17 - Dendrogram for the weighted linkage of the pre-trained word2vec model

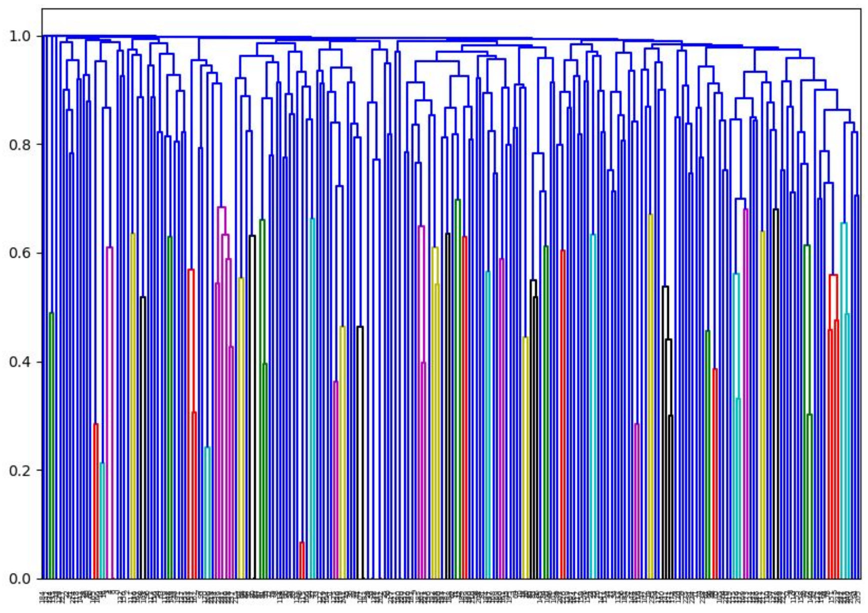


Figure 18 - Dendrogram for the weighted linkage of the the model that uses tf-idf embedding

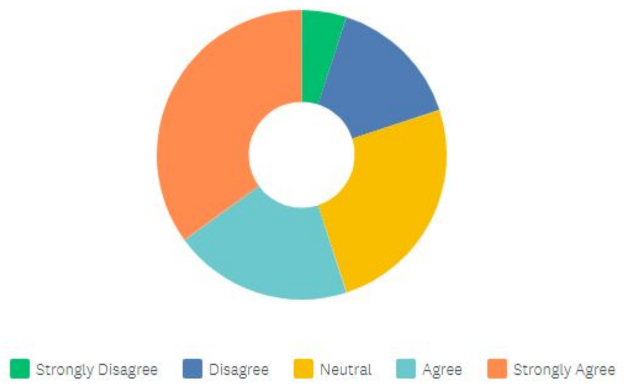
As it can be seen in Figure 17 and Figure 18, the cluster arrangements are suitable for clustering. The data points are clearly separable from each other since the distances between possible clusters are far enough. Therefore, we decided to continue evaluating the clusters that are produced with the embedded vectors that are generated by tf-idf model and the pre-trained word2vec model since they have a potential to be reasonable. After analyzing the dendrograms and some tuning, we decided to determine the distance thresholds for the agglomerative clustering to be 0.8 for both models.

5.2 Survey

This section is for representing the survey mentioned in the Evaluation subsection of the Methods section. Figures 19, 20, and 21 illustrate the survey results for the clusters [See Appendix C] obtained by the model that uses tf-idf embedding. Figures 22, 13, and 24 illustrate the survey results for the clusters [See Appendix D] obtained by the model that uses pre-trained word2vec embedding.

Related news headlines were grouped together.

Answered: 20 Skipped: 0

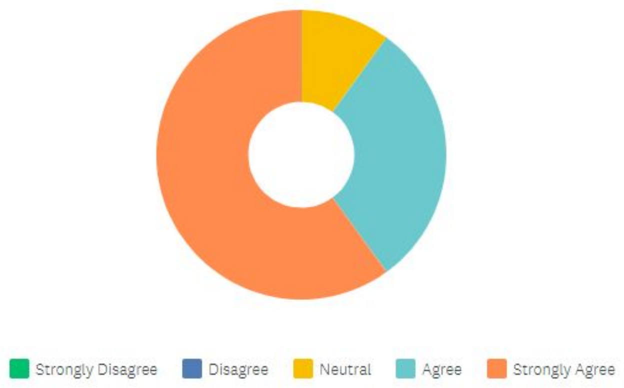


	STRONGLY DISAGREE	DISAGREE	NEUTRAL	AGREE	STRONGLY AGREE	TOTAL	WEIGHTED AVERAGE
(no label)	5.00% 1	15.00% 3	25.00% 5	20.00% 4	35.00% 7	20	3.65

Figure 19 - Survey question 1 results for the model that uses tf-idf embedding

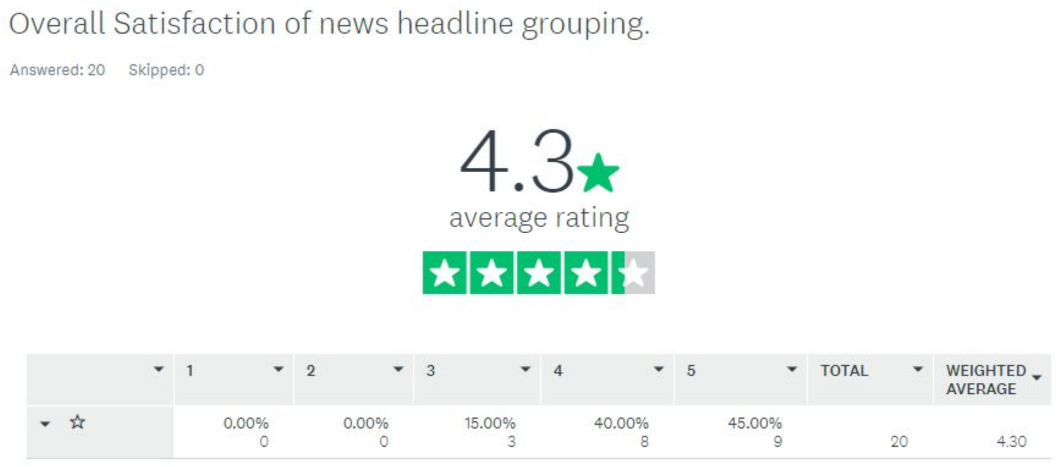
Unrelated news headlines were in separate groups.

Answered: 20 Skipped: 0



	STRONGLY DISAGREE	DISAGREE	NEUTRAL	AGREE	STRONGLY AGREE	TOTAL	WEIGHTED AVERAGE
(no label)	0.00% 0	0.00% 0	10.00% 2	30.00% 6	60.00% 12	20	4.50

Figure 20 - Survey question 2 results for the model that uses tf-idf embedding



Overall Satisfaction of news headline grouping.

Answered: 20 Skipped: 0

4.3★ average rating

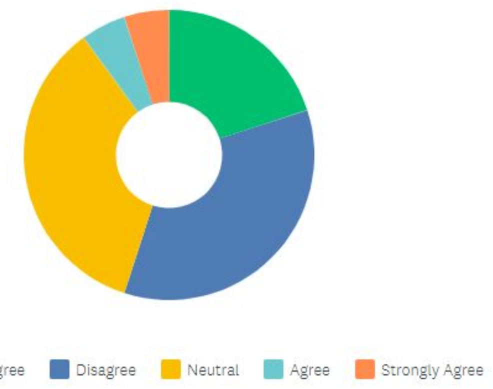
	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	15.00% 3	40.00% 8	45.00% 9	20	4.30

Figure 21 - Survey question 3 results for the model that uses tf-idf embedding

Figures 19, 20, and 21, show the result of the survey conducted using the clusters of the model with the tf-idf embedding. The survey was given to 20 selected people and their answers were collected anonymously. By looking at the survey results we can gain an insight into the completeness and the homogeneity of the clusters. Question 1 gives insight into the completeness and question 2 gives insight into homogeneity. Immediately we can say the homogeneity of the clusters is better compared to the completeness of the clusters. When the v-measure score is calculated from the survey results we get a v-measure score of 4.03. However, when we look at the overall satisfaction score given by the users we can see that the overall satisfaction score is 4.30. By comparing the overall satisfaction score and v-measure score, we can say that users give more importance to the homogeneity of the clusters compared to the completeness of the clusters. So homogeneity score should be more important than completeness score when evaluating the clusters.

Related news headlines were grouped together.

Answered: 20 Skipped: 0

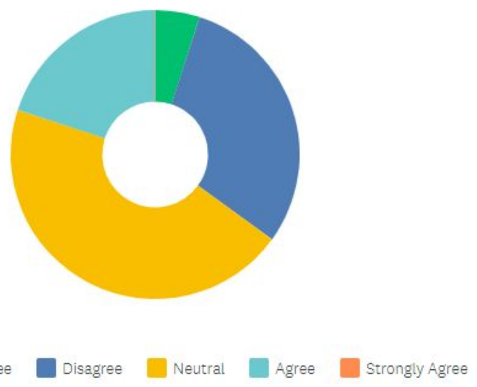


	STRONGLY DISAGREE	DISAGREE	NEUTRAL	AGREE	STRONGLY AGREE	TOTAL	WEIGHTED AVERAGE
(no label)	20.00% 4	35.00% 7	35.00% 7	5.00% 1	5.00% 1	20	2.40

Figure 22 - Survey question 1 results for the model that uses pre-trained word2vec embedding

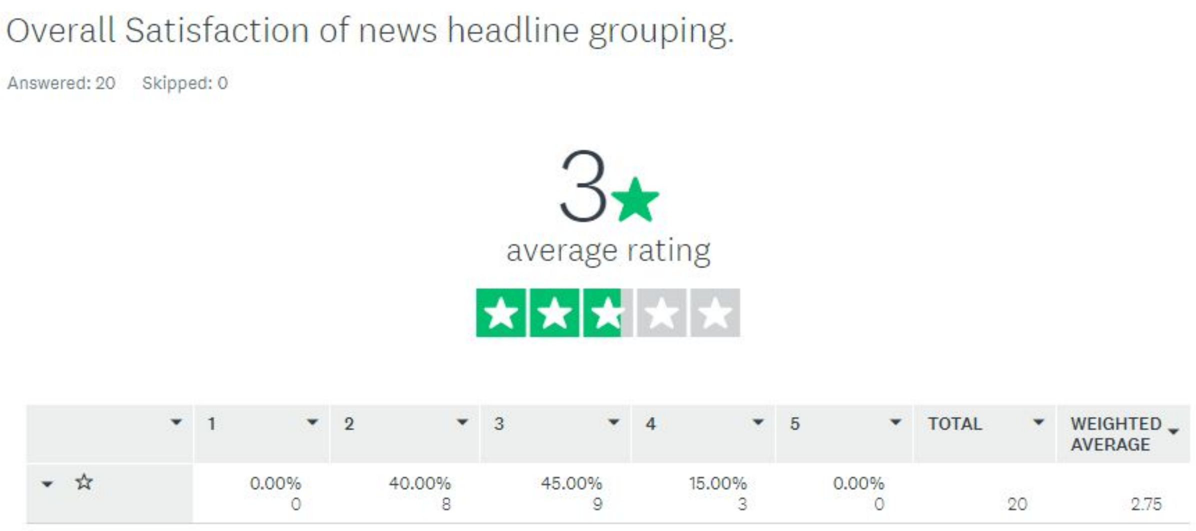
Unrelated news headlines were in separate groups.

Answered: 20 Skipped: 0



	STRONGLY DISAGREE	DISAGREE	NEUTRAL	AGREE	STRONGLY AGREE	TOTAL	WEIGHTED AVERAGE
(no label)	5.00% 1	30.00% 6	45.00% 9	20.00% 4	0.00% 0	20	2.80

Figure 23 - Survey question 2 results for the model that uses pre-trained word2vec embedding


Overall Satisfaction of news headline grouping.

Answered: 20 Skipped: 0

3★ average rating

	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	40.00% 8	45.00% 9	15.00% 3	0.00% 0	20	2.75

Figure 24 - Survey question 3 results for the model that uses pre-trained word2vec embedding

Figures 22, 23, and 24, show the result of the same survey conducted using the clusters of the model that uses pre-trained word2vec embedding on the participant of the first survey. We can again say the homogeneity of the clusters is better compared to the completeness of the clusters. However, this time difference is smaller compared to the first clustering. When the v-measure score is calculated from the survey results we get a v-measure score of 2.58. However, when we look at the overall satisfaction score given by the users we can see that the overall satisfaction score is 2.75. The results of this survey again suggest that homogeneity is a more important score compared to completeness for this clustering.

The model that uses tf-idf embedding performed better in completeness score, homogeneity score, v-measure score, and overall satisfaction score compared to the model that uses pre-trained word2vec embedding. We can comfortably say the tf-idf is the better embedding method for this system. The fact that the homogeneity score of the resulting clustering is extremely high for the cluster created using tf-idf embedding suggests that users will be satisfied with the clustering results.

5.3 Testers

In order to test the rest of our system, we asked our friends and families helps. We frequently gave latest build of our frontend applications to them to use. We then received their feedbacks on every detail and issue on the application. These feedbacks varied from UI bug reports, to news story grouping satisfaction to, recommendations on news sources. We believe that this was the perfect way for us to test our platform, with real users.

6. Maintenance Plan and Details

6.1 Information Retrieval Subsystem

It is important to re-mention that the performance of the Information Retrieval Subsystem is based on several outside Python Libraries. Therefore, until we develop our own modules, the maintenance of this subsystem will mostly be based on performing regular tests on the libraries we use and communicating with the authors if necessary. We are aware that being dependent on outside sources could produce unpredictable results and problems and affect the user experience negatively. To note, libraries like Twint and Boiler Pipe are less stable when compared to the other libraries used. As these two libraries are our main tools for retrieving the information that we want to deliver to our users, we will mostly focus on these two libraries' tests.

6.2 Google Firebase Cloud Application

Cloud functions are developed locally and afterward they are deployed to Google's servers so that they can be linked to the Firestore and Real-time Databases. Sometimes, Google informs its users that they need to redeploy their functions, therefore our maintenance plan is to regularly deploy different instances of our cloud functions and test their performance on several temporary databases.

6.3 Front-end Application

Because our front-end application is developed using Flutter, we only need to maintain a single codebase for android, ios, and web. This is one of the biggest reasons behind our decision to choose Flutter over other methods. Additionally, we are using a stable version of flutter which reduces the possibility of any bugs or exploits that can disrupt the user experience. And, because Flutter is backed by Google, we are confident that no breaking changes will be implemented in the stable version in the future that will affect backward compatibility. Thus, updates to Flutter and dart will be most likely unproblematic in the future. Our maintenance plan for the front-end applications is to improve user experience and implement new features to our application as we go on.

6.4 Clustering Subsystem

The news streams non-stop. Even though the incremental clustering keeps running constantly, the batch clustering should be performed frequently as well since concept drifts occur very often. However, it is not possible to set a time period for the batch clustering. A human should follow the stream of the news with respect to their contexts and should detect the concept drifts. When the current clusters are no longer usable because of the drifts, the batch clustering should be applied. In order to apply the batch clustering accurately, firstly, the dataset should be composed carefully. The datasets should contain the headlines within the concept that stand as long as possible. Secondly, the dendrograms should be analyzed and a proper distance threshold should be determined in each run of the batch clustering. In other words, the evaluation procedure that is followed in this paper should be applied for different datasets. In short, this system is useful and efficient for analyzing and comparing relatedness between the news but also requires effort to provide maintenance.

7. Other Project Elements

7.1 Consideration of Various Factors

Implementation Factor

For the development process of the NLP and Machine Learning algorithms of the Newspector, Python programming language and various different libraries were used. These libraries are very recent and not stable enough to use it with ease. We had to do a lot of research with a very small amount of documentation. Some of the libraries and methods required an extreme amount of computing power which was not possible for us to get a hold on. This lack of documentation with the combination of everchanging unstable libraries in the field made it extremely hard for us to implement our system.

For the Information Retrieval Subsystem we are again dependent on outside libraries and it is the main problem with this subsystem. Although our results from these libraries and tools seem pretty reliable, we are aware that in the long run we can not rely on outside sources. Due to our unsuccessful attempts for accessing the Twitter Developer API we had to use a web scraper to fetch the latest news from Twitter.

In the end we successfully constructed a working stable system for us to use in our application. However, we are aware that even the most advanced web scrapers can sometimes be unreliable, the same applies to NLP libraries as well. Therefore, our major plans are to gain access to the developer API, implement NLP systems by ourselves, and create an even more stable and reliable system.

Economical Factor

The Newspector is planned as a free-to-use platform with zero subscription fees. So implementing the application with the minimum amount of expenses was a hard task. We needed to research various ways to store data on the cloud. We also needed to research ways of deploying functionality to the cloud in order for us to run our server-side code. Finally, we needed to find open-sourced free solutions to our NLP needs. After thorough research, we decided to use Firebase for almost all our server-side needs as it was cheap and had a pay as you go approach. Developing our applications around Firebase pricing required different solutions to simple problems such as reading and writing to a database. We also found various methods, libraries, tools, etc. for our NLP needs for free thanks to the great open-source community behind machine learning. In the end, we successfully created a system that requires a minimal amount of expenses to run.

Political Factor

Since only “reliable and trusted” news sources should be selected to be used to gather news stories, and the users should not feel that these news sources are biased towards some political view, we decided to include as much variety into our news sources as possible. We research twitter accounts that share breaking news, and how they share it, whether we can use it or not. After our research, we picked the most reliable and diverse news sources that we could have chosen. Additionally, we treat every news source in our application equally, there are no special conditions for any news sources in our application. No political views were involved during the selection process and we implemented a rating system to the news sources so that users of our application can decide, and vote for themselves about news sources without our influence.

Privacy Factor

Since personal data protection is an important issue and there are regulations about it such as General Data Protection Regulation (GDPR) [34] by European Union (EU), we decided to not store any user-specific information in our databases. We do not collect any information about our users other than which newsgroups they follow. Even though, we can collect information about our users such as their preferences from the stories that they have read and use that information to target ads onto our users and increase our profits, we decided that user privacy is more important to us than increased profits.

Pandemic Factor

Most of our development process was during the global COVID-19 pandemic [35]. This drastically impacted our development process. We were unable to meet up in person and work as a fully functioning team. Additionally, the entire group members were under constant fear due to this pandemic. Our regular work schedules, plans were all changed, we had no access to our usual work areas and each other. We continued our development process from home with Zoom meeting, WhatsApp communication, phone calls, etc. However, because of how we dived the project and design each subsystem in our project to be worked on simultaneously by multiple people, we were able to mostly outcome the difficulties that the pandemic caused.

Table 1. Various factors and their effects.

Factors	Effects
Implementation Factor	6/10
Economical Factor	4/10
Political Factor	2/10
Privacy Factor	4/10
Pandemic Factor	9/10

7.2 Ethics and Professional Responsibilities

Developing software as a group is a challenging task on its own and continuing this development process without encountering a significant problem in a year-long project makes it even more challenging. To achieve an untroubled development process, communication between group members becomes key. Each member has to act like professionals, treat each other as equal, and be aware of the responsibilities that they have to each other. Even though each member has their own schedule and other responsibilities, they have to equally contribute to the project and manage their time to do so. We had no problems with working as a group. Even though our schedules were quite busy, we always had the time to communicate and work as a team.

Newspector will use data gathered from web scraping by reading the HTML of the website. Web Scraping can cause legal problems and the website being scraped can sometimes block access if it detects a bot is extracting the data. Because we gather our data from twitter, and scraping twitter is a very common task in the programming community, we had no problems scraping the data. However, we are still unsure if there are any ethical or professional problems with how we use the data gathered from twitter. We believe that we do not break ant ethical or professional rules as we are giving references to the original tweet and we use scraped material in the intended way as the original owner wanted.

Data such as which news stories the user reads, which newsgroups the user follows, or what is the favorite news source of the user is can be gathered and stored to improve the usability and the effectiveness of the Newspector. However, this information is not collected without the consent of the users, and if the user gives their consent for data gathering, the collected data is not associated with users directly and is kept anonymously to increase the privacy of users. Additionally, any information that is collected is not shared with any 3rd parties.

Any software, framework, algorithm, etc. that will be used in the development of the Newspector is licensed and credited accordingly. Additionally, we tried to use open-source software as much as we can to support and contribute to the open-source community.

Finally, because Newspector aims to increase the awareness of the newsreaders and try to reduce bias and prejudice by suggesting other perspectives and sources for the user to read, there can't be any bias in the program. The developers of the Newspector will not accept any incentives to favor one newspaper or the other.

7.3 Judgments and Impacts of Various Contexts

Not Including Sentiment Analysis in Final Product

Sentiment analysis of news content may be a problematic issue if the result is to be published to millions of users. Providing wrong sentiments could harm various social group's values and may create a misunderstanding that can result in a deviation in people's perception of Newspector. As we are dependent on an outside source for our sentiment analysis, it is too risky to provide this information to a lot of people. Therefore, we have decided to suspend this feature until we are certain about our results on analysis.

Table 2. Not Including Sentiment Analysis in Final Product Judgement with various impacts from different contexts.

	Impact Level	Impact Description
Impact In Global Context	Medium	We are aware the impacts of wrongful or controversial results of the sentiment analysis on global event. We do not want our platform to have a bias towards any part of the world.
Impact in Economic Context	Low	We know that providing a feature like sentiment analysis would draw more people to our platform, thus increasing profits. However, without a reliable sentiment analysis, we could not risk it.
Impact in Environmental Context	None	-
Impact in Societal Context	High	We want no one to feel like the sentiment analysis is wrong or offending to some part of the society.

Using Only News Headlines for Comparison

News headlines are different kinds of sentences when compared to the everyday life sentences people use. It is important to know that the news does not have a target audience as they are made for the whole community and they should be understandable by different social groups in the society. Therefore, the parts of speech in a news headline are very specific and most of the time it is very easy to understand the whole content with just a few words. Because of this, we have decided to make our content comparisons based on the headlines instead of the whole articles, which prevents getting undesired results as articles could contain advanced words and hidden phrases due to the flow of the whole content.

Table 3. Using Only News Headlines for Comparison Judgement with various impacts from different contexts.

	Impact Level	Impact Description
Impact In Global Context	High	We wanted our newsgroups to be as accurate as possible. We know that regardless of the news source sharing the news story, news headlines are the single most important thing that defines a news story. Most of the information will be given in the headline. So using the headline was the right choice for us.
Impact in Economic Context	Low	Using only the headline is also beneficial in terms of cost. Extracting the articles from each news source 100% accurately is a very costly task. And almost 100% accuracy is needed while extracting the news story as any irrelevant information will impact our clustering algorithms.
Impact in Environmental Context	None	-
Impact in Societal Context	None	-

Not Implementing Possible Unreliability Detection

Detecting varying information published on the same content requires a lot more effort than we taught. Even news having quantifiable information is hard to deal with as numbers in a breaking news are able to change as time progresses. Additionally, the sources that we use in our application are mostly verified sources or at least they have thousands of followers etc. Therefore, we thought that we could involve our users in the process of detecting unreliable information provided in a news. The users are able to provide feedback on the content of each news displayed in the application, so that our team would be able to make a detailed analysis on the news and remove the news if anything is misleading in it.

Table 4. Not Implementing Possible Unreliability Detection Judgement with various impacts from different contexts.

	Impact Level	Impact Description
Impact In Global Context	None	-
Impact in Economic Context	High	Detecting varying information on similar news stories requires a lot of effort and it is extremely costly to implement.
Impact in Environmental Context	None	-
Impact in Societal Context	High	Deciding for other people whether a news story is unreliable or not seemed presumptuous. Additionally, we thought that our users might not always agree, thus like our "unreliable" tags. So giving people all the resources so that they themselves can decide seemed the best option.

Not Storing User's Activity Data

Many applications collect information about their users such as their preferences and activities and use that information to target ads onto their users and increase their ad profits. This is especially true for our application. We have access to a lot of information on our users, which news stories they are reading, which topics that they are interested in, which newsgroups they are following, which news source they value the most, etc. We could use this information to show targeted ads to our users. However, we decided that user privacy is more important to us than increased profits and decided that we will not collect user data and use it for advertisements..

Table 5. Not Storing User's Activity Data Judgement with various impacts from different contexts.

	Impact Level	Impact Description
Impact In Global Context	None	-
Impact in Economic Context	High	Even though we could improve our income by targeting ads by using user information, we decided that increased income via reduced trust with our users was not for us.
Impact in Environmental Context	None	-
Impact in Societal Context	High	We are the developers of this platform but more importantly we are users of this and many other platforms. We value our own privacy and the privacy of our users. We believe that making user privacy our top priority is the right choice for our platform.

7.4 Teamwork and Peer Contribution

The project is divided into 3 main sections. The front-end applications, the information retrieval system, and the natural language processing system. These sections are distributed to 3 members equally. Each member was responsible for 1 section with other members helping out accordingly.

Front-end Application

Mainly developed by Ahmet Ayrancioglu with occasional help from Deniz Dalkilic.

Information Retrieval System

Mainly developed by Deniz Dalkilic with occasional help from Ahmet Ayrancioglu.

Clustering and Natural Language Processing

Mainly developed by Kaan Gonc.

Each section of the project was a tough challenge to tackle. All sectioned were developed in parallel and approximately completed around the same time witch each other. Each member was responsible for their own section and each member completed their section successfully. Each section was equally important for the project to function correctly. We had no issues regarding teamwork or work distributions and we believe that each member contributed to the project equally. Each person claimed leadership for their own section and provided other team members with needed updates, information, documentation, etc. It was fulfilling to work on this project as a team.

7.5 Project Plan Observed and Objectives Met

First of all, we need to mention that we have started our process with unrealistic requirements for our project which could have taken an enormous amount of time for a group of three programmers to accomplish. Therefore, with the feedback of our supervisor, we have slightly changed our requirements at each step so that our application would be more achievable and more efficient in the limited amount of time we have. The most important change was analyzing the news headlines instead of the articles for our clustering algorithms. The reason for this is that not all of the news had their own articles and a news headline was enough to identify a specific action taken by a specific actor and it was much easier to find the news having the same content. Another important plan was to find out possible unreliable information in the news and provide it to the user, however this was beyond the capability of the tools we used and we have decided to leave this detection of possible misleading information to our users. Our users are able to report any news with a proper comment so that we could control the reported news on whether it has misleading information or not. Finally, we have decided to remove the plug-in application from our plans as it would require a detailed analysis of the News Source websites and the format they share their news within the HTML. Therefore, we turned our focus mainly on the cross platform application that we have developed, which can be used in both IOS, Android and Web when the necessary arrangements are made.,

It is important to mention that we did not give up on our initial requirements that we have decided when we first started thinking about this project but rather left them as a future work as they will require a lot more effort on analysis and implementation. We believe, as a group of three people, we have accomplished more than we dreamed and came up with a nice project that could continue to improve and be in the market for a long time.

7.6 New Knowledge Acquired and Learning Strategies Used

First of all, we need to mention that none of the group members had any experience on the topic of Natural Language Processing. It was a completely new field for us to work on, therefore we can say that our development process mostly involved learning and applying new techniques and using new tools and technologies. We researched, tested, and used state of the art NLP libraries and techniques which were mentioned in the Development/Implementation Details section.

In addition to NLP, we had the chance to test, use, and learn various different tools, libraries, and techniques. Below are the tools, libraries, products, techniques, etc. learned through the development of the Newspector. We already mentioned what we learned and used in the implementation detail section of the report. So this section will contain only a list of tools, algorithms, libraries, frameworks, etc. we have learned while designing and developing this project and the learning strategies we used in order to learn and use these tools with ease.

Tools and Libraries

• Flutter • Dart • Firebase • Twint	• Google Cloud Language • Vader Sentiment • BoilerPipe

Learning Strategies

We took online classes from various sources such as youtube, udemy, and other websites. These online classes were mostly about Flutter, Firebase and Dart as most of the NLP tools we used were new and did not have online classes that talked about them.

We joined online forums and discussion channels. These forums helped us understand small details that were left out of documentations and classes. For example, we bumped into a problem about Firestore and was able to resolve the issue thanks to helpful people on the Firebase slack channel.

Unfortunately for us, because of the pandemic we were unable to go to any technical talks about the technologies we used. However, we listened to talks and conferences about some of the technologies we used on youtube.

We also had access to various different open-source projects that used some of the same technologies we used. We downloaded and built these systems to test and learn by example how some of the parts worked. Flutter had various open-source projects that we could look at.

8. Conclusion and Future Work

Obviously, our main plan is to see this application at the stores in the near future. However, there is some additional work that is needed to be done in order for us to publish a robust and fully functional application. First of all, our server applications are running in our local machines right now for debug and test purposes and we have to deploy them to a cloud platform that we can buy or rent. Then we need to create a fully functional build, and maintenance system using DevOps and tools designed for continuous integration and deployment. Additional arrangements should also be created for any crash situation so that our application and deployed servers can run and function properly all the time. Another issue is deciding the final user interface and we are planning to make surveys in order to select the most understandable and easy to use interface. We are aware that our target audience is everyone who uses a smart device therefore we should produce an interface that could address any social group in the society.

Overall, we are happy with Newspector's current situation and we believe that it will even improve more. We are confident that we provide an effective solution for the problem of accessing breaking news instantly and being able to prove them from multiple sources in daily life. Also we believe that integrating our application with social media and news sources' websites adds a lot more flexibility and effectiveness to our application as it provides instant access to alternative mediums to search for news and learn what is happening around the world at that time. Although a lot has changed from the first requirements step to the current state, we are happy with the final state of our application. We believe this iteration is the most useful iteration for a wider user base. We are also happy that we designed and developed a fully functional application in a short amount of time with a group of three people. Being able to develop an application which may impact millions of people is a very good experience even without publishing the final product.

References

[1] McCarthy, Niall. “Where Concern Is Highest About Fake News On The Internet [Infographic].” Forbes, Forbes Magazine, 12 June 2019, https://www.forbes.com/sites/niallmccarthy/2019/06/12/where-concern-is-highest-about-fake-news-on-the-internet-infographic/amp/.

[2] “Human Interface Guidelines - Design - Apple Developer.” [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/. [Accessed: 27-May-2020]

[3] “Material Design,” Material Design. [Online]. Available: https://material.io/design/. [Accessed: 27-May-2020]

[4] A. Leff and J. T. Rayfield, "Web-application development using the Model/View/Controller design pattern," Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference, Seattle, WA, USA, 2001, pp. 118-127, doi: 10.1109/EDOC.2001.950428.

[5] “Flutter - Beautiful native apps in record time.” [Online]. Available: https://flutter.dev/. [Accessed: 27-May-2020]

[6] “Dart programming language.” [Online]. Available: https://dart.dev/. [Accessed: 27-May-2020]

[7] “Cloud Firestore,” Firebase. [Online]. Available: https://firebase.google.com/docs/firestore. [Accessed: 27-May-2020]

[8] “Admin SDK Reference,” Firebase. [Online]. Available: https://firebase.google.com/docs/reference/admin. [Accessed: 27-May-2020]

[9] twintproject/twint. TWINT Project, 2020 [Online]. Available: https://github.com/twintproject/twint. [Accessed: 27-May-2020]

[10] “Cloud Natural Language,” Google Cloud. [Online]. Available: https://cloud.google.com/natural-language. [Accessed: 27-May-2020]

[11] C. J. Hutto, cjhutto/vaderSentiment. 2020 [Online]. Available: https://github.com/cjhutto/vaderSentiment. [Accessed: 27-May-2020]
“boilerpipe.” [Online]. Available: https://boilerpipe-web.appspot.com/. [Accessed: 27-May-2020]

[12] “boilerpipe.” [Online]. Available: https://boilerpipe-web.appspot.com/. [Accessed: 27-May-2020]

[13] “firebase_database | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/firebase_database. [Accessed: 27-May-2020]

[14] “cloud_firestore | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/cloud_firestore. [Accessed: 27-May-2020]

[15] “firebase_auth | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/firebase_auth. [Accessed: 27-May-2020]

[16] “google_sign_in | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/google_sign_in. [Accessed: 27-May-2020]

[17] “firebase_messaging | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/firebase_messaging. [Accessed: 27-May-2020]

[18] “flushbar | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/flushbar. [Accessed: 27-May-2020]

[19] “Google Fonts,” Google Fonts. [Online]. Available: https://fonts.google.com/. [Accessed: 27-May-2020]

[20] “google_fonts | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/google_fonts. [Accessed: 27-May-2020]

[21] “webview_flutter | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/webview_flutter. [Accessed: 27-May-2020]

[22] “Eva Icons - beautifully crafted Open Source UI icons for common actions and items.” [Online]. Available: https://akveo.github.io/eva-icons/#/. [Accessed: 27-May-2020]

[23] “eva_icons_flutter | Flutter Package,” Dart packages. [Online]. Available: https://pub.dev/packages/eva_icons_flutter. [Accessed: 27-May-2020]

[24] S. Qaiser and R. Ali, “Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents,” IJCA, vol. 181, no. 1, pp. 25–29, Jul. 2018, doi: 10.5120/ijca2018917395.

[25] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality.” [Online]. Available: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[26] M. Miháltz, mmihaltz/word2vec-GoogleNews-vectors. 2020 [Online]. Available: https://github.com/mmihaltz/word2vec-GoogleNews-vectors. [Accessed: 19-May-2020]

[27] "Sentence Similarity in Python using Doc2Vec," Kanoki, 07-Mar-2019. [Online]. Available: https://kanoki.org/2019/03/07/sentence-similarity-in-python-using-doc2vec/. [Accessed: 19-May-2020]

[28] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents" [Online]. Available: http://proceedings.mlr.press/v32/le14.pdf

[29] A. Griffiths, L. A. Robinson, and P. Willett, "HIERARCHIC AGGLOMERATIVE CLUSTERING METHODS FOR AUTOMATIC DOCUMENT CLASSIFICATION," Journal of Documentation, vol. 40, no. 3, pp. 175–205, Mar. 1984, doi: 10.1108/eb026764.

[30] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From Word Embeddings To Document Distances." [Online]. Available: http://proceedings.mlr.press/v37/kusnerb15.pdf

[31] R. R. Sokal and C. D. Michener, "A Statistical Method for Evaluating Systematic Relationships," The University of Kansas Science Bulletin, vol. XXXVIII, no. 22, pp. 1409–1438, Mar. 1958.

[32] "News Headline Grouping." [Online]. Available: https://www.surveymonkey.com/r/C6JDMJH. [Accessed: 20-May-2020]

[33] A. Rosenberg and J. Hirschberg, "V-Measure: A conditional entropy-based external cluster evaluation measure" [Online]. Available: https://www.aclweb.org/anthology/D07-1043.pdf

[34] "General Data Protection Regulation (GDPR) – Official Legal Text," General Data Protection Regulation (GDPR). [Online]. Available: https://gdpr-info.eu/. [Accessed: 27-May-2020]

[35] "Coronavirus." [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019. [Accessed: 27-May-2020]