

Topic generation using news article headlines in the Xray News iOS application

Gennadiy Shafranovich¹, Chris Avgerinos¹

 - to whom correspondence should be addressed

¹Twelve Products LLC - {gennadiy,chris}@xii.agency

Abstract

At Twelve¹, we developed an iOS application to allow users to easily follow news coverage on multiple topics, while being cognizant as to who published each news article and the outlet's political bias. This is the Xray News² application and as part of its development we needed a way to cluster³ news article headlines into topics (covering the same event or breaking news).

We created a pipeline that analyzes headlines and generates a keyword vector for each headline. Using the resulting keyword vectors we are able to calculate the similarities between headlines and cluster³ them into topics. Our pipeline operates on low power devices, without any need for processing done on back-end servers, and preserves the user's privacy. This allows it to be deployed as part of an iOS application to run on users' iPhones.

This write-up will describe the steps of the processing pipeline in detail and show how it is used to generate topic clusters of news articles.

1. Introduction

Here at Twelve¹ we strive to create products that solve a problem while being easy and pleasant to use. During the busy news environment of the covid-19 pandemic we took a look at our own doom-scrolling habits and thought: *There has to be a better way.*

To that end we created Xray News², an iOS application that lets you look at topics in today's news from multiple political perspectives. We think that even if you start reading about a topic from your individual place on the political spectrum, exploring what others have to say about it is useful to get a fuller picture.

News app strives to make it easier to consume news from outside your bubble.

Xray scans the news and groups all related stories about a subject in real time without any human interference.

Every story is color-coded and tagged according to its position on the U.S. political spectrum, making it easy to tell the underlying bias of everything you are reading.

The app is developed by Twelve and is currently on v0.2.0-7.

This is a developing story.

[Send us feedback](#)

[Terms & Conditions](#)

[Privacy Policy](#)




 Xray



Psaki Wore Mask

Psaki Says Kamala Harris Was Maskless Because She's 'Human' After Claiming She Had Worn A Mask

 Last updated 14 min. ago

Macron Le Pen

France's Le Pen Plays Down Far-Right Agenda to Broaden Appeal

 Last updated 21 min. ago

Hunter Biden

'Family, family, family:' Valerie Biden Owens defends brother 'Joey' and nephew Hunter

 Last updated 21 min. ago

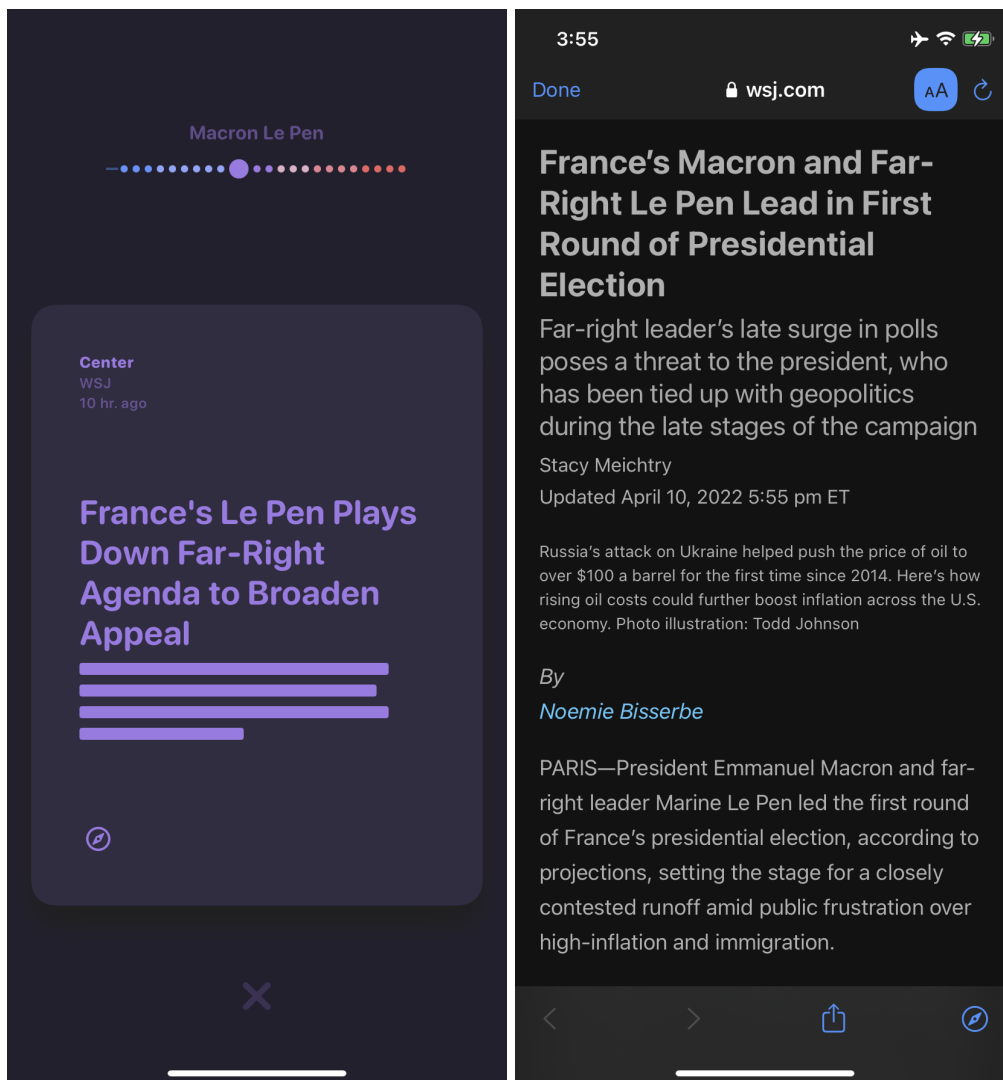


Figure 1 - Screenshots of the Xray News² iOS application.

One of the first challenges of building such an application is defining what is a *topic*. One definition of a *topic* is a set of articles, published by various news outlets, that all cover the same event.

Rewording this more formally:

What features of an article headline indicate that two (or more) articles are covering the same event?

We decided to only use article headlines to define topics since most news outlets do not provide the article's body in their RSS⁴ feeds. While some outlets provide abstracts to the article we found the presence of these to be so inconsistent as to not improve the clustering results.

The core of the problem is extracting topic identifying features from each headline, having a way of comparing sets of such features to find out if two (or more) headlines are covering the same event, and grouping news articles into topics using these comparisons.

1.1. Our contribution

We set out to find a solution to this problem so that we could package it up and deliver value to our users. What we found was that while there is plenty of prior work describing how to formulate topics from article headlines⁵⁶⁷⁸, most of the approaches require training of ML models⁹, high power computing, and/or centralized storage of both inputs and outputs of the process.

The following section describes the pipeline we developed to process article headlines and to cluster related ones into topics. We will describe how the raw text of each headline is transformed into a vectorized form, how these vectors are used to compare headlines to each other, and how these comparisons are used to cluster headlines into topics.

2. Method

2.1. Constraints of low power, low memory, and battery considerations

One of the reasons an off-the-shelf clustering solution was not used is our self imposed restriction to have all functionality run on a user's personal iOS device. While modern iPhones are extremely powerful processing devices, they are still a far cry from the power of dedicated hardware. In addition to processing speeds, we needed to comply with the operating system's constraints on how much memory and how much of the user's battery capacity our functionality can consume. This is only one of the self-imposed constraints, with another being the need to maintain user privacy. To this end, we decided not to have any processing at all happen away from the user's device.

Another chosen constraint was to only support devices running iOS version 15.0 or above. This restricts the application to only support newer iOS devices (iPhone 6S or newer), giving us an idea of the range of processing power and memory/battery capacity the user's device would provide us.

We also decided to set out a list of constraints on how fast certain pieces of the application needed to be, in order to provide a good user experience. These apply to the lowest end device we support (iPhone 6S on iOS 15+) and includes the headline clustering pipeline:

- Maximum time between the application being open and topic generation completion
 - 10 seconds
- Maximum time topic generation can take while the application is in background
 - 60 seconds
- The application also needs to achieve a level of responsiveness expected of all high quality applications on the iOS platform. This is subjective and was judged by surveying users who use iPhones as their main device and comparing their experience with other applications to ours.

Finally, we decided to concentrate on processing news and meta-data in English only. This allows us to use Apple's built in text processing utilities that have been tuned to the English language without having to find, test and optimize similar utilities for other languages.

2.2. Overview of the topic generation pipeline

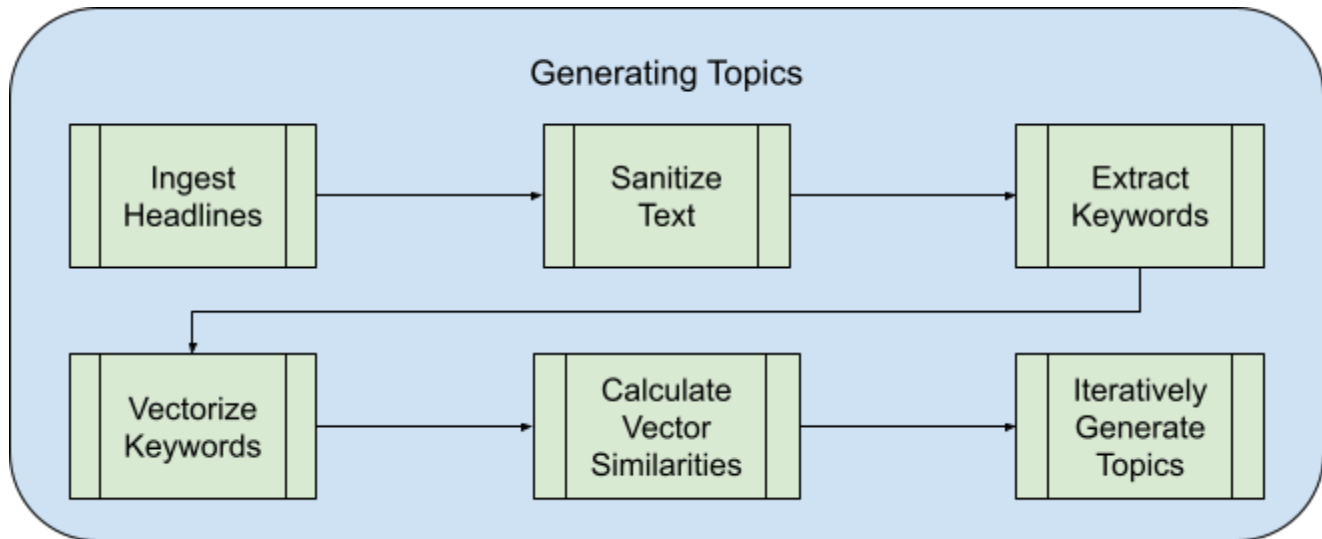


Figure 2 - Generating topics

2.3. Ingest news article headlines

News article headlines are ingested from various news outlets by parsing their RSS⁴ feeds (using the FeedKit¹⁰ iOS library). The contents of the feeds are refreshed (in parallel) using the user's network connection (either Wifi or cellular).

Some items are skipped during ingestion to remove noise in each outlet's feeds:

- Items older than three calendar days are skipped (based on their publication dates)
- Items disallowed by our static ingestion rules are skipped (e.g. video only items, listicles, etc.)
- Items with headlines longer than 150 characters are skipped

The above rules were obtained experimentally by looking at each outlet's feeds over a period of time and finding patterns that identify most items that are not news related.

2.4. Sanitize headline text

The raw headline text retrieved from an outlet's feed may contain various characters and patterns that prevent it from being processed correctly by the rest of the pipeline. Therefore the text of each headline is "sanitized" before it is processed further.

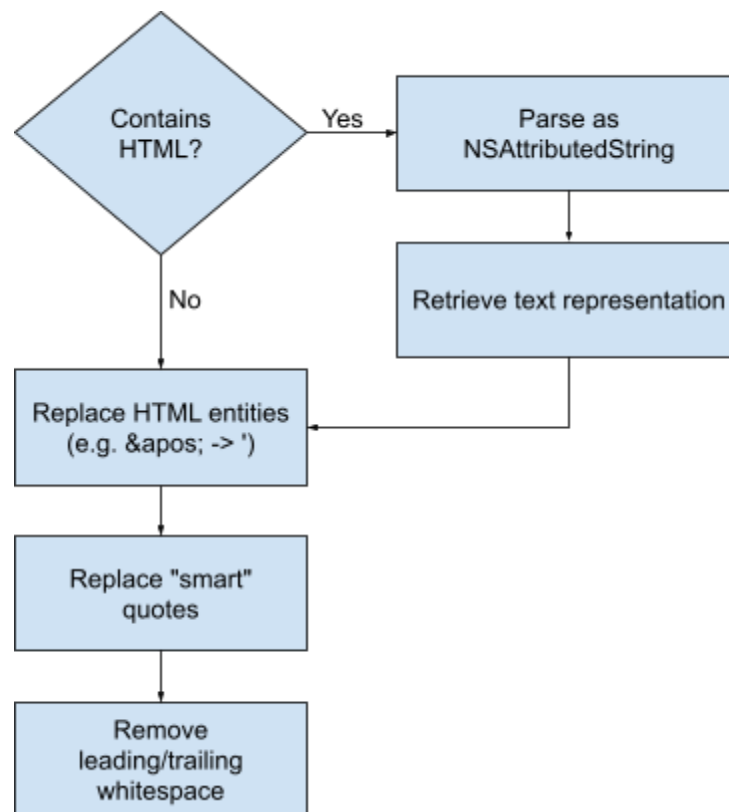


Figure 3 - News article headline text sanitization rules

If the headline text contains HTML, it is parsed using the NSAttributedString¹¹ utility and the underlying plain-text is extracted. Then we ensure that any HTML entities (e.g. &apos) are replaced with their plain-text counterparts (e.g. apostrophe character). Finally we replace smart quotes¹² with their ASCII equivalents and remove any leading/trailing whitespaces.

2.5. Extracting keywords from article headlines

The next step in the pipeline is to extract relevant keywords from each article's headline. The keywords are the "features" that will be used to describe the topic. Depending on the type of keyword (e.g. proper name vs noun vs verb) it may contain more than one word (e.g. "Joe Biden").

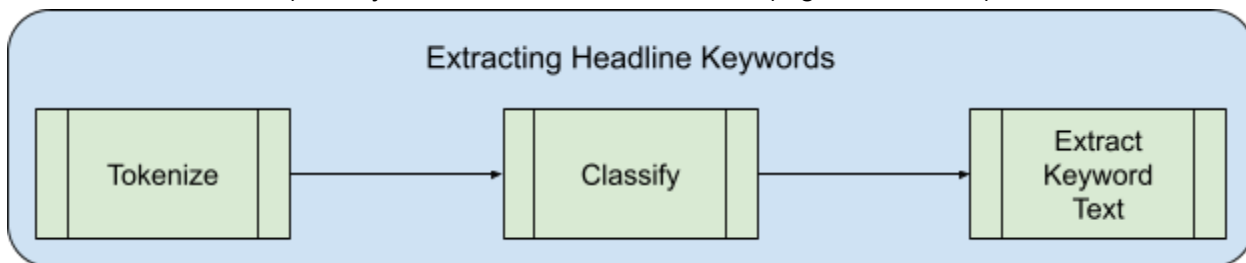


Figure 4 - extracting headline keywords

For each extracted feature (keyword) we need a few pieces of information:

- The type of the keyword
 - Person - a person's name (first, last, or full)
 - Organization - a name of an official organization (e.g. FBI, DHS, Supreme Court)

- Place - a name of a geographic location (e.g. New York City, Florida, Ukraine)
- Noun
- Verb
- Adjective
- Adverb
- Other - a catch-all bucket for keyword types that do not fit any other category
- The text of the keyword
- The position of the keyword within the headline (the keyword's index)

Thankfully, Apple offers native natural language processing (NLP) facilities for tokenizing¹³ text as well as an embedded natural language classifier model¹⁴. We apply both to each sanitized headline to obtain a list of keywords, with their original text, their lemmatized¹⁵ text, their name type (organization, place, person), and their lexical class (noun, verb, etc).

To detect the type of each keyword we use the its lexical class and name type (including "not-a-name") as follows:

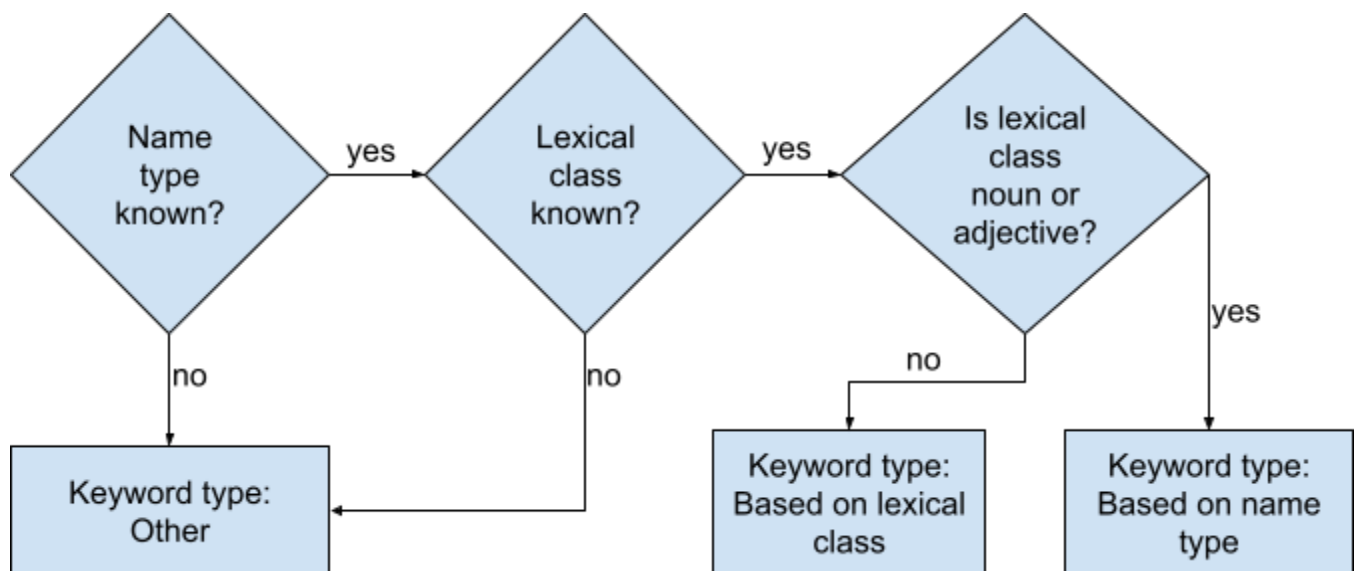


Figure 5 - Keyword type detection workflow

During experimentation we discovered a quirk of the embedded classifier utility where certain keywords are tagged as names when they definitely were not. We found that in 99% of such cases the lexical type (part of speech) was not a noun or an adjective. For example, the keyword "The First" was incorrectly classified as a name, with a lexical class of adverb. Experimentally we found that these classification errors occurred mostly when an article's headline had every word in it capitalized. We applied a check to ensure that the keyword's lexical class was either a noun or adjective, resulting in a high accuracy of predicting whether the name classification was valid or not.

Throughout the pipeline there is a need to compare keywords, and we do this using the keyword's text, which may contain more than one word (e.g. "joe Biden"). This may be different than the text of the keyword as it appears within a headline, to allow for comparisons that work regardless of the case of the keyword and other factors. To obtain the text of each keyword we first take its original text (as it appears in the headline) and remove any contractions from it. If the type of the keyword (see rules above) is one of the name types (person, org, or place) the keyword text is the result of the contractions

filter. This means that for names, we trust the original text within the headline (with contractions removed) and do not require any further processing.

For other types of keywords their text needs to be lemmatized¹⁵ before it can be used. Lemmatization¹⁵ is the process of representing the keyword by its lemma, or dictionary form, so that various inflections of a word are all treated as the same keyword in comparisons. For example, the words "run", "running" and "ran" all have the same lemma: "run".

Finally, the keyword text is checked against our embedded stop-word¹⁶ list. A stop-word¹⁶ list contains a set of words that do not on their own convey meaning (e.g. "and", "the", "them", etc). Only keywords that are not stop-words are used in further steps, all others are discarded (including empty strings after contraction removal). We sourced our stop-word¹⁶ list (with over 1k entries) from a public stop-word collection on github¹⁷.

2.6. Vectorizing headline keywords

For each headline, we create an N-dimensional vector with a magnitude value populated for up to 8 of the most important keywords within the headline. The maximum number of keywords per headline was chosen experimentally. Headlines that have less than a minimum (3) keywords are discarded from the process as well so that all resulting keyword vectors contain between 3 and 8 entries.

But, before we can assign a magnitude to each of these keywords, we need to decide which 8 keywords (per headline) are important, and their order of importance.

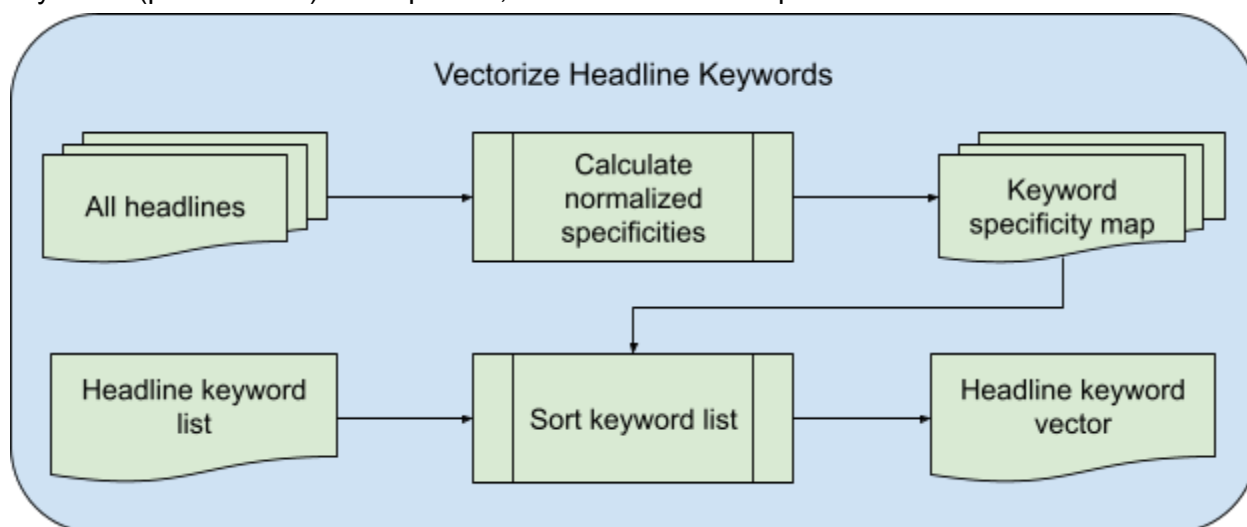


Figure 6 - Vectorizing headline keywords

2.6.1. Ordering headline keywords by their importance

Usually when processing a corpus of text the frequency of a keyword drives its importance (priority). This does not apply well to news article headlines since they are short and rarely contain repeated keywords. Article headline writers are specialists inside newsrooms and concentrate on being exact and concise, while still making you want to read the whole article.

With this in mind, we developed a keyword specificity measure, that is used in combination with the keyword's location within a headline, to gauge the importance of each keyword within a headline.

The keyword specificity measure is a normalized value between 0.0 and 1.0 where the higher the value is for a keyword the more specific it is judged to be. By "specific", what we mean is that the keyword is highly likely to describe a topic.

For example, the keyword "city" has a very low specificity value since topics covering all kinds of headlines would contain it. In contrast, the keyword "federal reserve" has a high specificity value since most headlines that contain it are likely to be talking about the same topic within our window for processing headlines (max age of a headline is 3 calendar days).

Since topics in Xray News² must contain a minimum number of articles each (minimum of 4), we do not need to calculate keyword specificity for any keyword that appears in less than 4 headlines.

For each keyword that appears in at least 4 headlines we gather the following data:

- All headlines where this keyword appears
- The list of all keywords (other than current) that appear in these headlines

Using these we generate the following statistics:

- Total volume
 - The total count (non unique) of keywords across all headlines where the current keyword appears, excluding the current keyword
 - For example if "Joe Biden" appeared with a total of 17 other keywords (non-unique, exclude current) across all headlines, the total volume is 17.
- Unique volume
 - The count of unique keywords across all headlines where this keyword appears, excluding the current keyword
 - For example if "Joe Biden" appeared with 9 unique keywords (excluding current) across all headlines, the unique volume is 9.

We produce a ratio of total and unique volume:

$$\text{Specificity ratio} = \text{keyword total volume} / \text{keyword unique volume}$$

A specificity ratio equal to 1.0 would mean that all keywords appearing with a particular keyword are unique, meaning that the keyword is likely not good at identifying a topic. The closer a specificity measure is to 1.0, the less likely it is to be good at identifying a topic.

In the example above, the keyword "Joe Biden" has a specificity of 1.88. This is close to 1.0 (unspecific), meaning that the other keywords it appears with do not have a high level of overlap, and likely describe different topics. Another example is the keyword "Ketanji Brown Jackson", with a specificity of 4.4. This is a signal that other keywords that it appears with have a high level of overlap, meaning that this keyword is good at identifying a topic by itself.

Overall this leads us to the following observation: the further away a specificity ratio is from 1.0 the better the keyword is at describing a single topic.

Because these ratios have a wide range we normalize the specificity measure across all keywords across all headlines as follows:

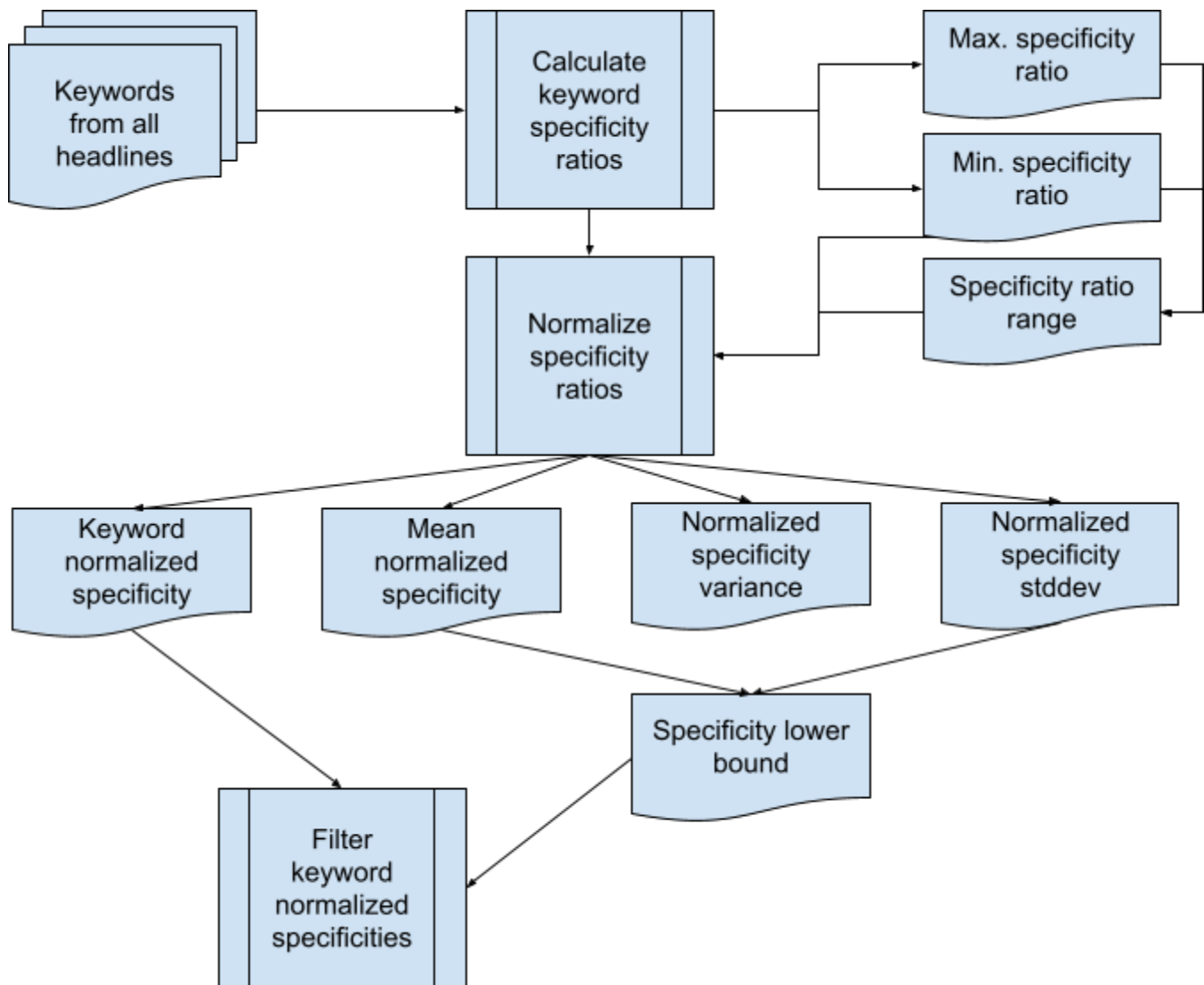


Figure 7 - Normalizing and filtering specificity ratios

First we calculate the specificity ratio for all unique keywords across all headlines in the corpus. That is followed by calculating the minimum, maximum, and range of the ratio values, which are used to calculate the normalized specificity value for each keyword:

$$\text{normalized specificity} = (\text{specificity ratio} - \text{min specificity ratio}) / \text{specificity ratio range}$$

Once each keyword has a normalized specificity, we filter out keywords that are so unspecific that they would just add noise to topic creation. We do this by calculating the mean, variance, and standard deviation of the normalized specificity for the population of keywords (across all headlines). We then define a lower bound of acceptable specificity as:

*Lower bound = mean normalized specificity - (1 * normalized specificity standard deviation)*

Any keyword with a normalized specificity below the lower bound is filtered out. Here are examples of a recent snapshot of keywords seen in news headlines on 2022-05-10. Some sample keyword specificities (from highest to lowest) are:

batali: 0.9813067629253163
...
ukraine: 0.6063745199788099
...
kavanaugh: 0.4330127018922193
...
illegal: 0.21821789023599233
...
global: 0.10721125348377959
...
issue: 0.08006407690254343
...
month: 0.07856742013183879
...
fed: 0.07784989441615207

For the same data sample the mean, standard deviation, and lower bounds are:

- Mean normalized specificity: 0.251724
- Normalized specificity standard deviation: 0.169677
- Specificity lower bound: 0.082047

In the above sample, the keywords "month" and "fed" are not included in the pipeline processes below because they are deemed to be unspecific. It is easy to see that these keywords are used in headlines covering many different aspects of the news and do not identify topics on their own.

2.6.2. Overall keyword importance

Using both the location of a keyword within its origin headline and the normalized keyword specificity value, we are now ready to prioritize and trim the keywords list for each headline. As a reminder, we are looking for top 8 keywords per headline, in order of their importance (best first).

For each headline, we take the list of processed keywords and apply the following prioritization:

- If keyword X's specificity is higher than keyword Y's, it has higher importance
 - If specificity is equal or missing, steps below are used
- If keyword X's type priority is higher than keyword Y's, it has higher importance:
 - First (highest): names (person, organization, place)
 - Second: nouns
 - Third: verbs, adjectives and adverbs
 - Fourth (lowest): other types
 - If type priority is the same, steps below are used

- If keyword X's earliest occurrence within its headline is earlier than keyword Y's, it has higher importance
 - Keywords appearing earlier in the headline have higher importance than those appearing later
 - NOTE: This final tie breaker introduces a bit of noise since headlines have varying lengths

Once we are able to sort a headline's keyword list using the above steps, we simply trim it to only include the 8 most important keywords per headline.

2.6.3. Generating the keyword vector

At this point each headline is linked to an ordered list of up to 8 keywords, ordered by their relative importance to that headline. We generate a vector from this keyword list based on the order of it's keywords:

- 1st (most important) keyword text = 1.0
- 2nd keyword text = 0.9
- 3rd keyword text = 0.8
- 4th keyword text = 0.7
- 5th keyword text = 0.6
- 6th keyword text = 0.5
- 7th keyword text = 0.4
- 8th (least important) keyword text = 0.3

The general formula for each entry in the vector is:

$$Nth \text{ keyword text} \rightarrow (1 - ((N - 1) * 0.1))$$

We use equally spaced vector magnitudes even though the keywords may have very different specificities, type priorities, etc. This prevents some keywords with a high specificity measure from outweighing the other keywords in the list, that may otherwise be used to differentiate one topic from another.

Here are some examples of headlines being processed into their vectors:

- Headline:
 - More human remains found in Lake Mead amid historic low water levels*
 - Sanitized headline:
 - More human remains found in Lake Mead amid historic low water levels*
 - Keyword list (original text in parentheses):
 - level (levels), lake (Lake), historic, human, mead (Mead), remains, water*
 - Vector:
 - level -> 1.0*
 - lake -> 0.9*
 - historic -> 0.8*
 - human -> 0.7*
 - mead -> 0.6*

- remains -> 0.5*
 - water -> 0.4*
- **Headline:**
Putin "doesn't believe he can afford to lose," CIA director says
 - Sanitized headline:
Putin "doesn't believe he can afford to lose," CIA director says
 - Keyword list (original text in parentheses):
 putin (Putin), lose, cia (CIA), director, afford
 - Vector:
putin -> 1.0
lose -> 0.9
cia -> 0.8
director -> 0.7
afford -> 0.6
- **Headline:**
Exclusive -- Sen. Marsha Blackburn: Biden Administration Trying to 'Police Speech' with Disinformation Board
 - Sanitized headline:
Exclusive -- Sen. Marsha Blackburn: Biden Administration Trying to 'Police Speech' with Disinformation Board
 - Keyword list (original text in parentheses):
 biden (Biden), disinformation (Disinformation), board (Board), sen (Sen), police (Police), exclusive (Exclusive), administration (Administration)
 - Vector:
biden -> 1.0
disinformation -> 0.9
board -> 0.8
sen -> 0.7
police -> 0.6
exclusive -> 0.5
administration -> 0.4

2.7. Calculating headline keyword vector similarities

We define a keyword vector similarity as a value between 0.0 (most similar) and 1.0 (least similar) for a source/target headline. Each headline in the corpus is treated as a source headline in turn. For each source headline, every other headline in the corpus (except itself, and those without enough keywords) are treated as target headlines in turn.

One optimization that was implemented within Xray News² is to only compare each unique pair of headlines only once. Since both headline A -> headline B similarity and headline B -> headline A similarity have the same value, we only need to process each unique pair of headlines once, leading to $((N-1) * (N - 2)) / 2$ comparisons in total, instead of $(N-1)^2$.

The following process is used to calculate the keyword vector similarity between source and target headlines:

- Calculate the cosine similarity¹⁸ between the source and target headline keyword vectors
 - Results in a measure between 0 (most similar) to 1 (least similar)
 - Optimized within Xray News² for high performance via pre-calculated vector magnitudes, using set overlap operations to retrieve the set of keywords forming the numerator, etc
- If the similarity is not within our minimum threshold (0.6) it is discarded
 - The threshold is adjustable and was obtained experimentally via subjective grading
 - The threshold is optimized for our chosen maximum number of keywords per headline (8), and would need to be adjusted and re-graded if that maximum is changed

Once the set of all similarities (that are within our chosen threshold) between all pairs of headlines is known, we are ready to generate topics. In the Xray News² application these results are generated iteratively (for all newly ingested headlines each time a refresh occurs) and stored. This allows for everything except actual topic generation to be pre-calculated in an efficient manner and re-used each time we generate topics.

2.7.1. A note on keyword vector similarities in practice

Take an example of having 1,000 headlines in the headline corpus that require similarity pre-calculation. Let's say they all have valid keyword sets and have a minimum number of keywords required (3). This leads to $1,000 * 1,000$ similarity comparisons, or 1 million possible similarities to store. Even if we only process similarities in an ordered manner (using unique identifier lexicographic ordering) we will need to perform $(1,000 * 999) / 2$ calculations, yielding just around 500k possible similarities.

The prospect of calculating and storing 500k to 1 million records on a mobile device is a bit daunting from a processing speed as well as memory pressure standpoint, so let's take a look at our threshold measure and what it means in real world usage. The likelihood of all 1k articles covering the same topic in the real world is near zero so we can assume that only a small fraction of the 1 million possible similarity pairs would actually have a similarity value at or below our 0.6 threshold. In practice we see less than 1% of the possible similarity pairs fall within the threshold, meaning that the 1 million possible records get reduced to around 10,000 records.

In addition, since news is published over time (and not all at once) the 1,000 source headlines do not appear in Xray News² all at once. We take advantage of this fact by processing newly ingested headlines only and removing any old data (older than 3 days) from the system. This yields an average of 500 to 1,000 new similarities from 20 to 50 new headlines per update batch, which is very manageable from a performance and user wait time perspective.

2.8. Iteratively generate topics

Before we dive into the process of topic generation itself, a quick note about why topics are generated but not stored within the Xray News² application. The reasoning behind this is that while article data and its similarity information is static, topics are more dynamic. Some article groupings may make sense when there are only 4-10 articles covering a topic and no longer make sense when there are 20+ articles covering the same topic. In the latter case, the topic may need to be broken down into more detailed topics, covering different aspects of the original story.

For example, coverage of a recent incident on the NYC subway¹⁹ started with breaking news items stating the few facts were known at the time. Coverage then shifted to talking about the victims of the incident, the search for the suspect of the crime, and how local politicians were responding to the crisis. While it made sense at first to group all the breaking news coverage into a single topic, it no longer made sense to do so as the coverage became more detailed and nuanced.

2.8.1. Seeding topics

To start the topic generation process every headline is considered to reside within its own separate topic. We use the headline keyword vector similarity measures to seed the distance measure between each topic pair. Since only a small fraction of headlines are actually similar (within our threshold), this yields a small in-memory data structure containing the topic distances.

2.8.2. Iteratively merging topics

We use the hierarchical clustering²⁰ (agglomerative) approach to go from our seed topic set to the final set of topics to show the user. The linkage criteria²¹ is the median²² of all cross headline (within topic) similarities, with each headline similarity being the cosine distance¹⁸ between the headline pair's keyword vectors.

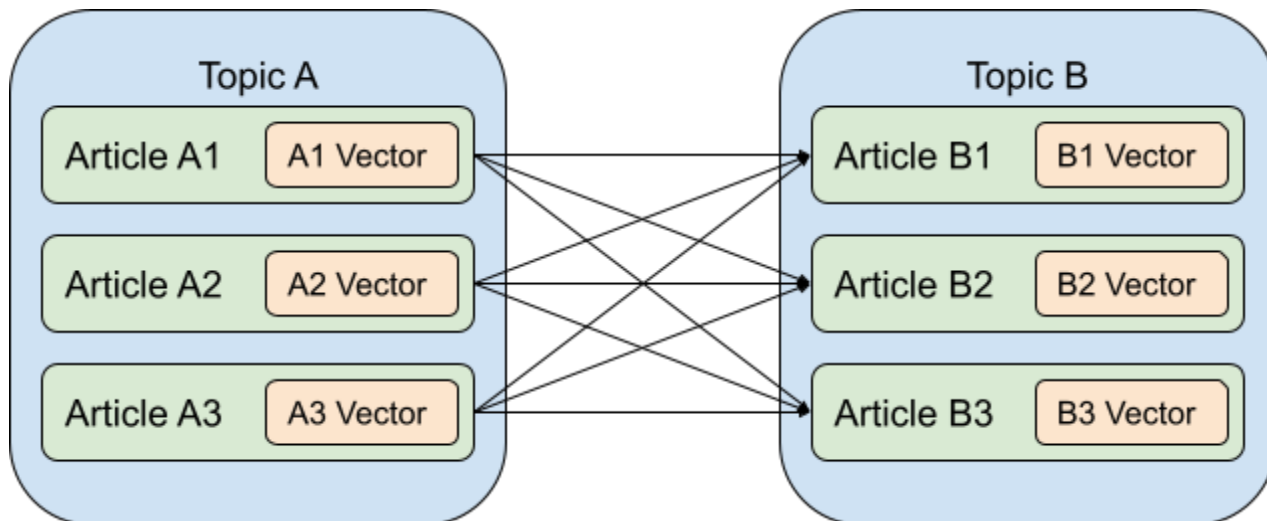


Figure 8 - Topic similarity calculation

The similarity between topic A and B in the example above is calculated as:

```
median(
    cosine distance (A1 vector -> B1 Vector),
    cosine distance (A1 vector -> B2 Vector),
    cosine distance (A1 vector -> B3 Vector),
    cosine distance (A2 vector -> B1 Vector),
    cosine distance (A2 vector -> B2 Vector),
    cosine distance (A2 vector -> B3 Vector),
    cosine distance (A3 vector -> B1 Vector),
    cosine distance (A3 vector -> B2 Vector),
    cosine distance (A3 vector -> B3 Vector),
```

)

We start the merge process by retrieving a pair of topics from the topic similarity data structure that has the minimum similarity (using the calculation above) that is still within our threshold (0.6). This pair represents two topics that are "most similar" within the current topic set. Since all known topic similarities must at least meet our threshold this means that the topics are known to be similar AND be the most similar.

Using the "most similar" pair, we form a new topic which contains all headlines from both of the topics of the pair. The two source topics are then removed from our tracking data structure. We need to do this since their individual distances to other topics are not representative of the distance of the merged topic to every other topic.

Next we need to calculate the distance of the newly formed topic to every other known topic. In order to reduce the number of computations required we will only consider topics that "can" be similar to the newly formed one. This is done by looking up all headlines that are similar to ones within the topic (using the same threshold of 0.6), and then finding the topics those headlines are contained in. The unique set of topics that this traversal yields contains the only topics that can have distances to the newly formed one that are within our threshold.

The same topic distance calculations are now applied to each target topic and the newly formed topic, updating the tracking data structure to take the newly formed topic into account.

We experimented with many linkage criteria and many topic distance measures but found that the median²² of all cross headline similarities yielded the best results.

The merge process above is repeated by finding the next "most similar" topic pair, merging it, and updating the similarity data structure until we can no longer find such a pair.

3.0. Results

The result of the iterative topic generation process is a set of topics, each containing a list of headlines (with their source article links). Once generated, the topic list is not stored permanently. We chose to re-generate the topics on demand (application open or user requested refresh) to have the most up to date topics to show the user.

Using the topic generation pipeline described here we were able to build a user experience around browsing each topic, seeing summaries of the articles behind each headline, and give the user the ability to visit the outlet's website to read the source article.

In the day to day usage of the application we see this topic generation approach run faster than the thresholds we set out at the start (both in foreground and background modes), allowing for a smooth user experience.

Acknowledgements

We would like to include our thanks to Sohini Chakraborty, Postdoctoral Fellow at the NYU School of Medicine, for her suggestion of publishing this paper and her help in preparing it. We would also like to thank John Edwards for his early work on the Xray News² concept while at Twelve¹.

Thanks is also due to Simon Fondrie-Teitler, Yakov Shafranovich, Johannes Krämer, and Karla Brkić for their comments and suggestions on this write-up.

Authorship contributions

John Edwards and Chris Avgerinos developed the original concept for the product and developed early prototypes. Gennadiy Shafranovich and Chris Avgerinos worked on adopting the concept as an iOS application, including the subjective grading and iterative development of the algorithm. Gennadiy Shafranovich was the principal software engineer on the project, including the optimization phase for mobile devices, integration with various third party libraries, and overall integration with iOS.

Conflict of interest disclosures

Gennadiy Shafranovich, Chris Avgerinos, and John Edwards were employed at Twelve¹ and are owners of the company, during the development of the Xray News² and its underlying algorithm. Twelve¹ has chosen not to monetize the algorithm described in this write-up and will not commercially benefit from it, only from the final product application.

Figures

Figure 1 - Screenshots of the Xray News application on an iPhone XR device showing grouped news topics, summaries of a single article, as well as the full article view on the publisher's website.

Figure 2 - Generating topics: ingest headlines, sanitize text, extract keywords, vectorize keywords, calculate vector similarities, iterative generate topics

Figure 3 - News article headline text sanitization rules:

- If the headline contains HTML, process via NSAttributedString¹⁴ to retrieve text
- Replace HTML entities with their counterparts
- Replace smart quotes¹⁵
- Remove leading/trailing whitespace

Figure 4 - Extracting headline keywords: tokenize, classify, extract keyword text

Figure 5 - Keyword type detection workflow

- If the name type of the keyword is not known, keyword type is Other
- If the lexical class of the keyword is not known, keyword type is Other
- If the lexical class IS NOT a noun or adjective, keyword type is based on the lexical class
- If the lexical class IS a noun or adjective, keyword type is based on the name type

Figure 6 - Vectorizing headline keywords:

- Obtain keyword specificity map: calculate normalized specificity values and build keyword specificity map
- Sort each headline's keyword list based on the keyword's specificity value, keyword type, and index within the headline
- Vectorize the sorted keyword list for each headline

Figure 7 - Normalizing and filtering specificity ratios:

- Calculate specificity ratios
- Obtain minimum, maximum, and range of specificity values
- Normalize specificity ratios across all headlines
- Calculate the normalized specificity mean, variance, and standard deviation
- Filter all specificities to only include those above the lower bound (mean - 1 x stddev)

Figure 8 - Topic similarity: median of cosine similarities of all article pairs

References

[1] *Twelve*. c2022. <https://xii.agency>

[2] *Xray - Pop your news bubble*. c2022. <https://xraynews.app>

[3] Wikipedia contributors. Document clustering. Wikipedia, The Free Encyclopedia. March 13, 2022, 06:13 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Document_clustering&oldid=1076837881

[4] Wikipedia contributors. RSS. Wikipedia, The Free Encyclopedia. February 26, 2022, 19:42 UTC. Available at: <https://en.wikipedia.org/w/index.php?title=RSS&oldid=1074165867>. Accessed April 27, 2022.

[5] Altuncu, M. Tarik, Sophia N. Yaliraki, and Mauricio Barahona. "Content-driven, unsupervised clustering of news articles through multiscale graph partitioning." arXiv preprint arXiv:1808.01175 (2018).

[6] Gu, Xiaotao, et al. "Generating representative headlines for news stories." Proceedings of The Web Conference 2020. 2020.

[7] Wubben, Sander, et al. "Clustering and matching headlines for automatic paraphrase acquisition." Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009). 2009.

[8] Laban, Philippe, Lucas Bandarkar, and Marti A. Hearst. "News headline grouping as a challenging nlu task." arXiv preprint arXiv:2105.05391 (2021).

- [9] Wikipedia contributors. Machine learning. Wikipedia, The Free Encyclopedia. May 3, 2022, 20:14 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1086035935
- [10] Nuno Diaz, FeedKit - An RSS, Atom, and JSON Feed parser written in Swift, (2012), GitHub repository, <https://github.com/nmdias/FeedKit>
- [11] Developer.apple.com. 2022. NSAttributedString | Apple Developer Documentation. <https://developer.apple.com/documentation/foundation/nsattributedString>
- [12] Wikipedia contributors. Quotation marks in English. Wikipedia, The Free Encyclopedia. April 11, 2022, 10:43 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Quotation_marks_in_English&oldid=1082104747
- [13] Developer.apple.com. 2022. Tokenizing Natural Language Text | Apple Developer Documentation. https://developer.apple.com/documentation/naturallanguage/tokenizing_natural_language_text
- [14] Developer.apple.com. 2022. Identifying Parts of Speech | Apple Developer Documentation. https://developer.apple.com/documentation/naturallanguage/identifying_parts_of_speech
- [15] Wikipedia contributors. Lemmatisation. Wikipedia, The Free Encyclopedia. May 12, 2021, 21:45 UTC. Available at: <https://en.wikipedia.org/w/index.php?title=Lemmatisation&oldid=1022854248>
- [16] Wikipedia contributors. Stop word. Wikipedia, The Free Encyclopedia. November 27, 2021, 16:19 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Stop_word&oldid=1057431778
- [17] Gene Diaz, Stopwords ISO, (2016), GitHub repository, <https://github.com/stopwords-iso/stopwords-en>
- [18] Wikipedia contributors. Cosine similarity. Wikipedia, The Free Encyclopedia. February 11, 2022, 16:54 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=1071254454.
- [19] Wikipedia contributors. 2022 New York City Subway attack. Wikipedia, The Free Encyclopedia. April 29, 2022, 03:04 UTC. Available at: https://en.wikipedia.org/w/index.php?title=2022_New_York_City_Subway_attack&oldid=1085213048.
- [20] Wikipedia contributors. Hierarchical clustering. Wikipedia, The Free Encyclopedia. April 27, 2022, 20:56 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Hierarchical_clustering&oldid=1085000321
- [21] Wikipedia contributors. Hierarchical clustering. Wikipedia, The Free Encyclopedia. April 27, 2022, 20:56 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Hierarchical_clustering&oldid=1085000321.
- [22] Wikipedia contributors. Median. Wikipedia, The Free Encyclopedia. April 23, 2022, 03:41 UTC. Available at: <https://en.wikipedia.org/w/index.php?title=Median&oldid=1084200350>.

