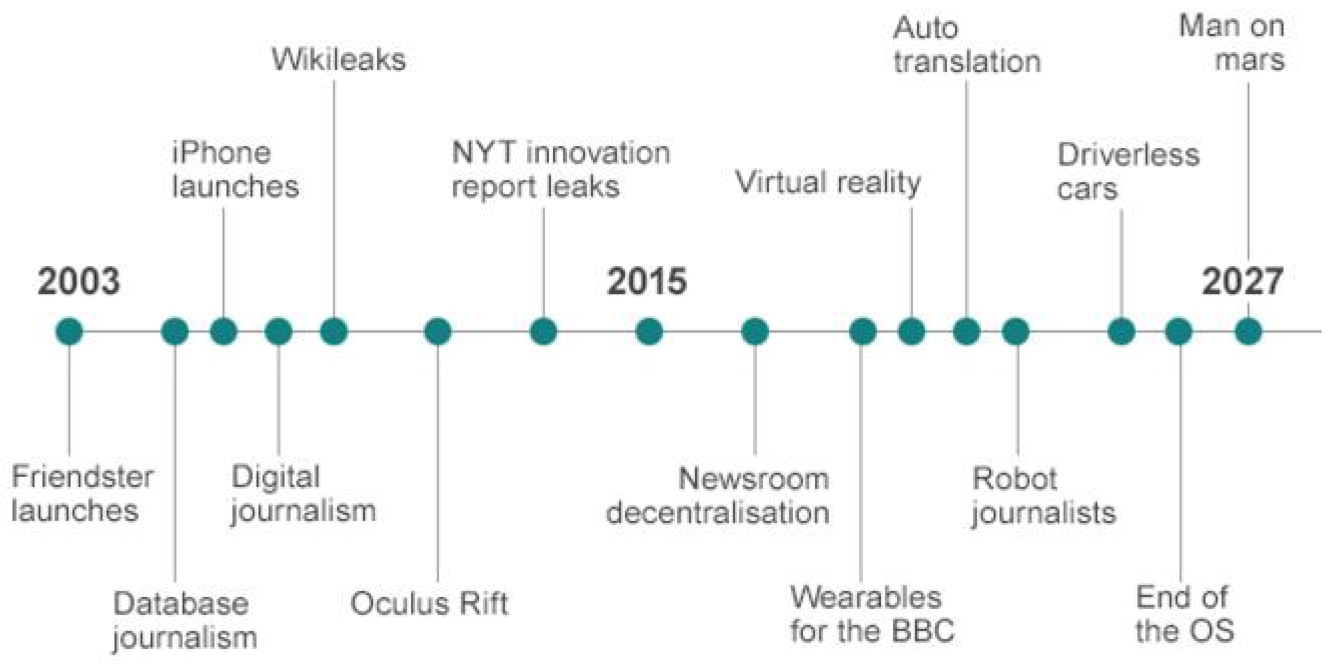
Auto⏳Timeline: Generate a high-quality timeline of any topic using Machine Learning!

Ankur Pandey
May 21, 2019 · 5 min read



In this age of the internet, we have access to lots of data and one such example is access to many news articles. With such a huge amount of data, it is easy to get confused. The mission of **UnFound** is to fight misinformation & all its cousins, & we believe that we believe that it is important to target information overload.

▶ Enter Timeline

Say we have a topic such as the ***World Trade Organization*** which has been evolved with time. One good way can be to look at its *timeline*. We will highlight our approach on how to generate timeline automatically using machine learning from news articles.

Problem Statement

Given some input word or phrase, figure out top ***n*** relevant events from news articles related to our input.

Dataset

To start any machine learning project we need data. For this problem, we harvested a corpus of news articles for our topics (say, *World Trade Organization*), which is the given input to generate a timeline.

Challenges in Auto Timeline generation

There hasn't been much work done in this timeline generation as it's a niche problem hence there are no end-to-end algorithms to do so. Some issues are mentioned here:

- Extracting *events* for a topic.
- Selecting *important* events for the timeline from all the events.
- Some of the most important events will appear more than once. To remove the redundant events is also a challenging task.

Proposed approach

As already mentioned, we don't have a complete algorithm to solve this problem, we will try to combine multiple techniques of NLP & statistics, and solve this problem by breaking it in *sub-problems*.

STEP 1: Query expansion

While looking for a given topic (say, ***Mamata Banerjee***), it is useful to expand the query to relevant words/phrases.

We used a **word2vec** model trained on unigrams, bigrams, & trigrams in our data- to get the synonym sets to expand the query. Thus, we'll get sets like these:

> *Query: 'Mamata Banerjee', Synonyms: {'Mamata didi', 'Trinamool Congress', ..}*

STEP 2: Event extraction

From the corpus first we need to pick candidate events and the date of occurrence of the event. Though event extraction is a very big field in itself, we tried different techniques like VerbNet to extract the events, but it turned out to be slow & bulky. Then we moved towards a simple and elegant approach to extract the events: which is to pick only those sentences which have a date or temporal words (like *today, tomorrow, hereafter,* etc)in it. This approach worked quite well as we got all the events in articles.

Examples:

> 1. ***Events:*** *On 23 January 2017, the amendment to the WTO Trade-Related Aspects of Intellectual Property Rights (TRIPS) Agreement marks the first time since the organization opened that WTO accords have been amended, and this change should secure for developing countries a legal pathway to access affordable remedies under WTO rule.*
>
> ***Date****: 23 January 2017*
>
> 2. ***Events:*** *Since it helped shape the General Agreement on Tariffs and Trade, the WTO's predecessor, in 1948, setting the rules for world trade, the United States has led every round of trade liberalization.*
>
> ***Date****: 1948*

Though we are able to pick almost every event from the articles, there are some obvious issues! For example, consider the topic "***Mamata Banerjee***", & an event extracted:

> *From the stage of 21 July Martyr's Day, Trinamool Congress' most important annual event, the chief minister called for a more united Opposition and announced the start of a statewide 'Quit India, BJP' programme from 9 August, the same day it was launched by Mahatma Gandhi back in 1942.*

The date 1942 does NOT correspond to the topic! We circumvented it by extracting events from an ***adaptive window*** near the possible timeline of our topic.

STEP 3: The relevance of events

Now, when we have lots of events, which are directly extracted from the articles, we may get some *not-so-important* events for our timeline. We use **KL divergence + embeddings** based approach.

Kullback-Leibler Divergence is a way of comparing two probability distributions. Embeddings, as used before, are a way to encode text as vectors. We combine both as follows:

```
KL Score:
```

```
1. create a distribution (relative term frequency of words) of all
articles concatenated together .
2. create a distribution (relative term frequency of words) of each
event.
3. take KL divergence between the articles distribution and events
distribution

Tag score:

1. create sentence embedding of each events.
2. concat tags to create a sentence and then create its embeddings.
3. use cosine similarity between each events and tags to get tag
score.

Final score:

combine both the score to get the final relevance score.
```

STEP 4: Duplicate removal & event diversification

Now that you have candidate events and the events are ranked according to their relevance. At this point, we need to solve the problem of **duplicate events.**

It occurs because two or more articles might have talked about the same event and you might have picked both as candidate events (in STEP 1). And the problem gets worse when those two events are described in a *different* way!

Duplicate removal can be a big headache, but that is when **clustering** comes in to rescue us. We can cluster all the event such that similar events fall into the same cluster.

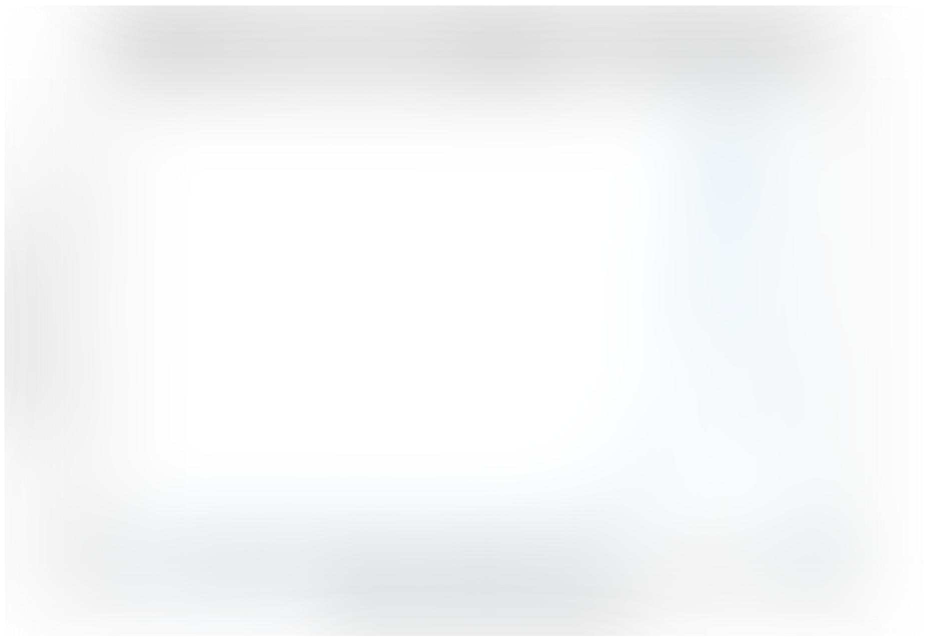
We ***bucket*** all events in a group (year, month, etc- depends on the use case). We first use a **hierarchical clustering algorithm** to find appropriate clusters (more on this later); and then apply a **diversification scheme** to ensure that unique events are selected as well the source of these events are as diverse as possible. Diversification is important because we at times want to diversify the source, authors, etc.

STEP 5: Top *n* events

By now, let's say you have 150 unique *good* events, but for the timeline, you need, say, only 20. How to pick those 20 events from 150 depends upon your use case.

One might pick just the top 20 **most relevant** events- but these events might end up being concentrated only in a small temporal window! This is a problem, because a typical timeline is all about the occurrence of events from beginning to end, and a good distribution of events is important!

Let's look at the KDE plot of the timeline for the topic "***Mamta Banerjee"*** for a better understanding:



The graph shows the distribution (KDE plot) of events date for the topic 'Mamata Banerjee'. The graph tells us that the events relevant to Mamata Banerjee started after 1950 (and any event before that are spurious). This is actually supported by the fact that Mamata Banerjee was born on 5 January 1955. You can see the peek after 2000 that depicts the political career of Mamata Banerjee (there is some mass in distribution around 2050 and before 1950 which comes as a result of smoothing by KDE plot).

In order to solve this, we follow this rule:

```
Expected events= N
Number of events from a year t= (probability mass function of the
event distribution at t)*(N)
```

. . .

This was by no mean the best possible approach- and there is a lot of work in progress. Still, it gave us surprisingly high-quality results on par with timelines published in top news sources. Our biggest learning is that in the case of ill-posed problems like this, the way one breaks down the problem in constituents is the most important factor.

Timeline is internally used to power **UnFound news app**. We'll release a public API very soon. Stay tuned.

Machine Learning Timeline Ai At Unfound NLP Misinformation