Information Retrieval Project - Topic Modeling

Jesper van Stee - 20205019
Stefan Chiru - 20202124

December 2020

1 Introduction

Currently there are many news articles published daily. It is safe to say that people are not interested in reading every article that is published, different people have different interests. In this report we discuss the creation of a topic modeling system that will help address the problem of assisting readers to find articles they are interested in by generating common topics based on news article content. The code of the project, as well as results can be found at:
`https://github.com/J-esper/IR-Topic-Modeling`

2 Data

The data used for this project is a `csv` file containing news article information. The dataset contains 141 585 rows with 9 columns:
`id, title, publication, author, date, year, month, url, content`.

We are mostly interested in the `content` of the news articles for topic modeling, therefore we disregard all other columns except the `id`.

3 Method

In order to create the Topic Modeling System we use Latent Dirichlet Allocation (LDA) a description of LDA can be found in section 3.2.

Two different models were used in our attempts:

- Bag of Words LDA (BOW LDA)
- Term Frequency-Inverse Document Frequency LDA (TF-IDF LDA)

The difference between the two models is that Bag of Words is a simple vector with a count of all the words in a document. TF-IDF uses weights that contains information about more important words and lesser important words. Words

like *and*, *the*, and *are* are very frequent. The more a word appears across the documents, the lesser the importance becomes.

3.1 Word representations

TF-IDF and BOW vectors count the exact spellings of terms in a document. So texts that restate the same meaning will have completely different vector representations if they spell things another way for grammar reasons, or use synonyms. This messes up search engines and document similarity comparisons that rely on counts of tokens. [2]

3.2 What is LDA

The definition of LDA from the original paper: "A generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document."[1]

3.3 Preprocessing

To create a more uniform dataset we applied some preprocessing steps.

3.3.1 Lemmatization stemming

We labeled each of word, with their lemma, and then we processed these new tokens instead of the original words. This lemmatization approach keeps two similar spelled words together in our analysis, but not necessarily words with similar meanings. And it definitely failed to pair up most synonyms. Synonyms usually differ in more ways than just the word endings that lemmatization and stemming can deal with. Even worse, lemmatization and stemming sometimes erroneously lump together antonyms, words with opposite meaning. The end result is that two chunks of text that talk about the same thing but use different words will not be "close" to each other in our lemmatized TF-IDF vector space model. And sometimes two lemmatized TF-IDF vectors that are close to each other aren't similar in meaning at all. Even a state-of-the-art TF-IDF similarity score, such as Okapi BM25 or cosine similarity, would fail to connect these synonyms or push apart these antonyms. Synonyms with different spellings produce TF-IDF vectors that just aren't close to each other in the vector space.

3.3.2 Stop words

We remove certain stop words. The first set of stop words are are provided by `nltk.corpus` for the English language.
We extended the original dataset of stopwords that NLTK comes with, with the following: *from, subject, re, edu, use, not, would, say, could, _, be, know, good, go, get, do, done, try, many, some, nice, thank, think, see, rather, easy, easily, lot, lack, make, want, seem, run, need, even, right, line, even, also, may, take, come.*
These were added because they were not initially included and do not carry meaningful information for sentences in the articles.

3.3.3 Regular Expressions

Another preprocessing step we applied is removing certain characters from the content. We do this by filtering with regular expressions.

3.3.4 n-grams

We are not filtering only the too short or too long words, we are also filtering on the allowed pos (Part of Speech) tag : NOUN, ADJECTIVE, VERB, ADVERB
We are also building bi-grams and tri-grams in order to find the statistically most significant words
Using n-grams enables our algorithm to know about "ice cream" as well as the "ice" and "cream" that comprise it.

3.4 Implementation

3.4.1 PyMC

We have followed the following tutorial: `http://alfan-farizki.blogspot.com/2015/07/pymc-tutorial-3-latent-dirichlet.html`. We have used PyMC (python package that contains multiple types of distribution) to create the model. The algorithm was extremely slow, as for $10\dot{0}00$ articles the time to run was longer than 10 minutes on a six-core CPU. There was no parallelization or any kind of optimization.

3.4.2 Gensim

In LDA models, each document is composed of multiple topics. But, typically only one of the topics is dominant. LdaMulticore : This module allows both LDA model estimation from a training corpus and inference of topic distribution on new, unseen documents. The model can also be updated with new documents for online training.
`simple_preprocess(doc, deacc=False, min_len=2, max_len=15)`

Converts a document into a list of lowercase tokens, ignoring tokens that are too short or too long.

3.5 Visualization

3.5.1 Tensorflow Embedding Projector

`https://projector.tensorflow.org/`
We wanted to explore UMAP embeddings of the dataset using Embedding Projector from Tensorflow. The result is an interactive 3D view where we can search labels and tags on the data. We load up our word2vec vectors.

- UMAP
- PCA
- t-SNE

4 Experiments

We ran several experiments. First we will discuss the results of running the Bag of Words and TF-IDF LDA with 50 000 of the rows of the dataset. It takes a very long time to run one experiment so we opted to reduce the size of the dataset a little bit. We randomly sampled from the dataset with a random seed set.

4.1 Bag of Words

We ran the BOW LDA with three different number of topics: 10, 20, and 50. We first ran with 20 topics and 50 000 rows. However, this still takes quite a long time so for comparing the 10 and 50 number of topics we run it with 10 000 rows.

4.1.1 Results

First the results of the LDA with 20 topics.

Topic	**0.0**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**	**6.0**	**7.0**	**8.0**	**9.0**
Count	2560	2995	2974	2190	1199	1548	2574	5246	3525	2674
Topic	**10.0**	**11.0**	**12.0**	**13.0**	**14.0**	**15.0**	**16.0**	**17.0**	**18.0**	**19.0**
Count	1177	445	3410	915	517	2194	2488	3230	4649	3478

Table 1: Documents per topic using Bag of Words for number of topics: 20

We see that many of the articles are classified under the 20 different topics. Some seem to appear more often than others. Figure 1 shows the wordclouds of the first 9 topics.

Next we compare using 10 topics against using 50 topics. Table 2 for the results of using 10 topics and Table 3 for the results of using 50 topics.

Topic	**0.0**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**	**6.0**	**7.0**	**8.0**	**9.0**
count	1144	1361	1214	1018	1100	521	379	408	2139	710

Table 2: Documents per topic using Bag of Word with number of topics: 10

Topic	**0.0**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**	**6.0**	**7.0**	**8.0**	**9.0**
Count	53	690	103	45	819	24	329	201	123	42
Topic	**10.0**	**11.0**	**12.0**	**13.0**	**14.0**	**15.0**	**16.0**	**17.0**	**18.0**	**19.0**
Count	57	36	113	63	358	85	29	51	77	80
Topic	**20.0**	**21.0**	**22.0**	**23.0**	**24.0**	**25.0**	**26.0**	**27.0**	**28.0**	**29.0**
Count	122	82	48	326	71	22	149	44	26	139
Topic	**30.0**	**31.0**	**32.0**	**33.0**	**34.0**	**35.0**	**36.0**	**37.0**	**38.0**	**39.0**
Count	253	293	106	217	102	102	45	185	95	64
Topic	**40.0**	**41.0**	**42.0**	**43.0**	**44.0**	**45.0**	**46.0**	**47.0**	**48.0**	**49.0**
Count	63	163	618	604	280	1327	435	401	27	207

Table 3: Documents per topic using Bag of Word with number of topics: 50

We see that both still have articles being categorized for each of the topics. However, using 50 topics shows that a few topics have many more articles occur than other topics, for example topic 45 has 1327 documents, while the second most occurring one has 690 documents. This is less prevalent when using 10 topics, however there topic 8 is still dominant.

The terms for dominant topic 45:
`trump, campaign, voter, people, candidate, election, vote, republican, party, political`

The terms for dominant topic 8:
`trump, vote, campaign, people, election, voter, republican, state, candidate, year`

We see that these keywords are closely related, so we can assume that the majority of articles are about this topic and these results seem valid.

Figure 1: Wordcloud for first 9 topics of Bag of Words LDA

4.2 TF-IDF

4.2.1 Results

We notice that using TF-IDF all articles seem to be classified under mostly one topic, scanning the resulting `csv` shows some strange connections between the topic and actual article content, so we opted to not continue with the TF-IDF LDA for further results.

Topic	**0.0**	**1.0**	**2.0**	**3.0**	**4.0**	**5.0**	**6.0**	**7.0**	**8.0**	**9.0**
Count	51	1	113	0	2	11	62	0	0	3
Topic	**10.0**	**11.0**	**12.0**	**13.0**	**14.0**	**15.0**	**16.0**	**17.0**	**18.0**	**19.0**
Count	1	2	1	3582	11	1	0	4714	2	41431

Table 4: Documents per topic using TF-IDF

4.2.2 Word2Vec

Word2Vec learns the meaning of words merely by processing a large corpus of unlabeled text. This unsupervised nature of Word2vec is what makes it so powerful. The world is full of unlabeled, uncategorized, unstructured natural language text. All words in our corpus will be represented by numerical vectors, similar to the word-topic vectors, the topics mean something more specific, more precise comparing with TF-IDF or BoW.
Word vectors learn word relationships based on the training corpus. If our corpus is about finance then our "bank" word vector will be mainly about businesses that hold deposits. If your corpus is about geology, then our "bank" word vector will be trained on associations with rivers and streams.
We first created the vector of embdedings using the Tensorflow tutorial on Word2Vec. The vector and the metadata can be found in the repository.
We have trained a Word2Vec model using *Gensim* on the *NewsDataset* and compared it with *Google News 300* dataset. And the results were suprisingly good for our model.

In terms of size, our vector of embeddings had a size of about 480 MB while the Google vector had a size of more than 1.2 GB. A few comparisons:

- our model most similar(positive=['congress'], topn=5)
  - parliament, 0.61428
  - congress, 0.594975
  - Parliament, 0.55161
  - parliament, 0.5368
  - delegation, 0.49542
- Google's model most similar(positive=['congress'], topn=5)
  - congresses, 0.617
  - Congress, 0.615
  - plenary_session, 0.605
  - plenary, 0.595
  - muktamar, 0.530

Figure 2: Wordcloud for the first 4 topics of TF-IDF LDA

Word2vec could have been used by selecting the top word for each topic and selecting the top word by similarity in from the vector of embeddings, this way we could extend the topic modelling with similar words. Extending the space of topics could make the topics more human readable (resulting in a higher *perplexity* score).

4.3 Compute model Perplexity and Coherence score

Coherence score and perplexity provide a convenient way to measure how good a given topic model is.
In order to decide the optimum number of topics to be extracted using LDA, topic coherence score is always used to measure how well the topics are extracted: CoherenceScore= $\sum_{i<j} score(w_i, w_j)$ where w_i,w_j are the top words of the topic. Lower the perplexity better the model. Higher the topic coherence, the topic is more human interpretable. Calculate per-word likelihood bound, using the chunk of documents as evaluation corpus.
This shows that the Bag of Words outperforms the TF-IDF, and using more

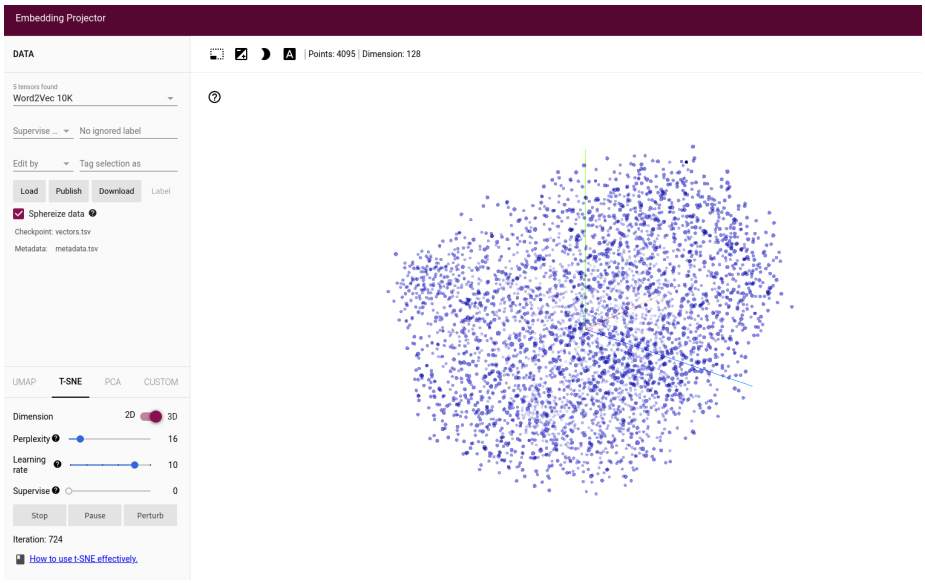

Figure 3: Word embeddings visualization

Model	10k TF-IDF	10k BOW	50k BOW
Perplexity	-12.5233	-12.6768	-11.6771
Coherence Score	0.5807	0.6133	0.6436

Table 5: Perplexity and coherence score for different models

data improves the coherence.

4.4 Top 100 per topic

In the `top_100_50k.csv` file the top 100 document ID and score of each topic (20 topics) are stored. We see that every topic has a document with a very high score related to that topic.

5 Conclusion

With the experiments we found that using Bag of Words LDA yields better results than Term Frequency-Inverse Document Frequency LDA. The documents are spread out better over the topics. Using a larger number of topics seems to cause a bigger disproportion between the number of articles in each topic. Lastly, we find that using a more data increases the coherence score.

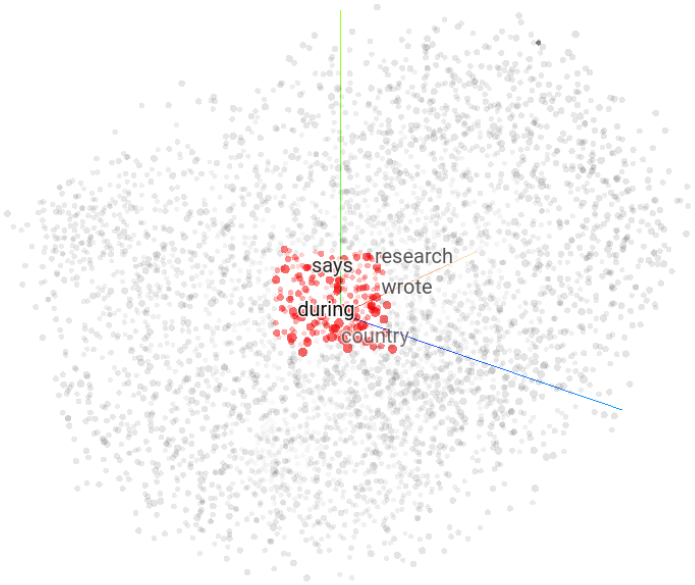

Figure 4: Selection of word embeddings inside the 3D model T-sne

5.1 Future Work

One of the biggest challenges was the time it takes to run the experiments on our setups. For later research we would like to be able to use a larger dataset, this would most likely improve our results significantly.

5.1.1 TOP2VEC: DISTRIBUTED REPRESENTATIONS OF TOPICS

"A useful topic model should find topics which represent a high-level summary of the information present in the documents. Each topic's set of words should represent information contained in the documents. For example, one can infer from a topic containing the words warming,global,temperature, and environment, that the topic is global warming. We define topic modeling to be the process of finding topics, as weighted sets of words, that best represent the information of the documents. " [3] This paper comes with the idea of having distributed representations of topics. The author found that LDA and PLSA (Probabilistic Latent Semantic Analysis) one weakness of these methods is that the number of topics is discrete, therefore "each topic produced by these methods is a distribution of word probabilities. As such, the highest probability words in a topic are usually words such as the,and,it and other common words in the language. These common words, also called stop-words, often need to

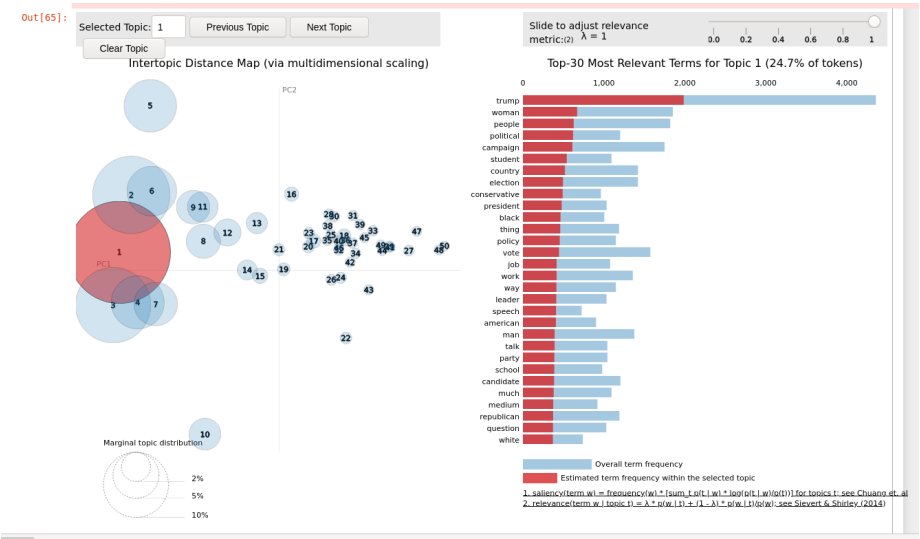

Figure 5: Topic distribution using pyLDAvis

be filtered out in order to make topics interpretable, and extract the informative topic words. Finding the set of stop-words that must be removed is not a trivial problem since it is both language and corpus specific; a topic model trained on text about dogs will likely treat dog as a stop-word since it is not very informative."[3] In essence top2vec is a combination of vectors of embedded topics, documents, distances. How top2vec works:

- Create semantic embedding: using neural networks it creates embeddings, not only on the words and distance between them but on the distance for words and documents. In this case it is almost the same as word2vec only that it counts one more feature.

- Find Number of Topics: to find the optimal number of topics for each document it uses Hierarchical Density-Based Spatial Clustering of Applications with Noise, even though this would have two problems, when the document vectors are sparse and because of a spar

- Low Dimensional Document Embedding: the author chose to use UMAP, mainly because, by nature, this algorithm allows options for clustering (sparse or dense, i.e global structure preservation or local structure preservation). It does this by chosing a k representing the number of neighbors, as the metric, cosine similarity is used.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3, null (3/1/2003), 993–1022.

[2] Hobson Lane, Cole Howard, Hannes Max Hapke. Manning Publications Co. - Natural Language Processing in Action

[3] Dimo Angelov. Top2Vec: Distributed Representations of Topics `https://arxiv.org/pdf/2008.09470.pdf`