Topic Modelling Techniques using Linear Algebra

May 2020

1. Abstract

In machine learning and natural language processing[1], a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modelling[2] is a frequently used unsupervised text-mining tool for discovery of hidden semantic structures in a text body. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently: "dog" and "bone" will appear more often in documents about dogs, "cat" and "meow" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, in a document that is 10% about cats and 90% about dogs, there would probably be about 9 times more dog words than cat words.

The "topics" produced by topic modelling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is. In the age of information, the amount of the written material we encounter each day is simply beyond our processing capacity. Topic models can help to organize and offer insights for us to understand large collections of unstructured text bodies. Originally developed as a text-mining tool, topic models have been used to detect instructive structures in data such as genetic information, images, and networks. They also have applications in other fields such as bioinformatics.

2. Keywords

Machine learning, Natural Language Processing, Topic Modelling, Text Mining, Unsupervised Learning, Statistical Model, Semantic Structure, Dimensionality Reduction, Feature Extraction, Information Retrieval, Eigenvectors, Eigenvalues, Orthogonality, LSA, SVD, NMF

3. Introduction

We have used the 20 News Group Dataset[3], originally made by Ken Lang for his paper on filtering news from the net. It is an extremely popular dataset using for a variety of natural language processing tasks from text classification to text clustering and even topic modelling.

The dataset, which has also been published on Kaggle as well as other renowned machine learning repositories such as the UCI Machine Learning Repository and is even a part of scikit-learn, contains almost 20,000 news documents across each of its 20 news categories. Some of the topics include hardware, motorcycles, electronics, space, sales, politics, and sports.

Our goal is to find the underlying topics within this dataset and try and obtain all the words corresponding to each topic in each news group, which will then be compared to the group's topic for accuracy measures. To perform topic modelling, we need to extract appropriate features so we can pick the words that offer most relevance to each topic and these words will indicate the underlying topics in each set of documents. Our goal is to use the two most powerful linear algebraic techniques in topic modelling namely, Latent Semantic Analysis (LSA) and Non-Negative Matrix Factorization (NMF) and compare the results of these two approaches, view the topics identified by each algorithm and perform a simple cluster analysis. Both algorithms work on the principle of matrix factorization. LSA is a powerful statistical technique that performs matrix factorization on a document-term or term-document matrix using truncated Singular Value Decomposition (SVD) to reduce its dimension and obtain the features which provide us with the most singular values. NMF factorizes a matrix into two non-negative matrices where each of these matrices hold the relevance of words to topics and features.

4. Latent Semantic Analysis – LSA

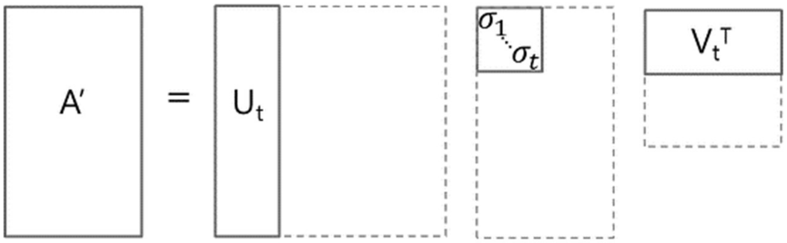
4.1 Linear Algebra Background

Theorem 1. *The singular value decomposition of an* ***m x n*** *real or complex matrix* **M** *is a factorization of the form* $\mathbf{USV^T}$*, where* **U** *is an* ***m x m*** *real or complex unitary matrix,* **S** *is an* ***m x n*** *rectangular diagonal matrix with non-negative real numbers on the diagonal, and* **V** *is and* ***n x n*** *real or complex unitary matrix. If* **M** *is real,* **U** *and* $\mathbf{V^T}$ *are real orthonormal matrices.*

$$\mathbf{M = U\ S\ V^T}$$

Theorem 2. *The diagonal entries* **S** *are known as the singular values of* **M**.

Theorem 3. *The truncated singular value decomposition of a matrix* **M** *is performed by taking only the* ***t*** *column vectors of* **U** *and* ***t*** *row vectors of* $\mathbf{V^T}$ *corresponding to the* ***t*** *largest singular values in* **S** *and discarding the rest of the matrices.*



4.2 Mathematical Background

Let $\mathbf{X}$ *be a matrix where* $\mathbf{X}_{\mathbf{ij}}$ *describes the occurrence of term* **i** *in document* **j** *(this can be, for example, the frequency).* $\mathbf{X}$ *will look like this.*

$$\begin{array}{cc}
 & \begin{matrix} \phantom{x_{1,1}} & & \mathbf{d}_j & & \phantom{x_{1,n}} \\ & & \downarrow & & \end{matrix} \\
\mathbf{t}_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix}
\end{array}$$

$\mathbf{t}_\mathbf{i}$ *stands for term* **i** *and* $\mathbf{d}_\mathbf{j}$ *stands for document* **j**

Now a row in this matrix will be a vector corresponding to a term, giving its relation to each document.

$$\mathbf{t}_i^T = \begin{bmatrix} x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \end{bmatrix}$$

Likewise, a column in this matrix will be a vector corresponding to a document, giving its relation to each term.

$$\mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{i,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

Now the dot product $\mathbf{t}_\mathbf{i}^\mathbf{T}\mathbf{t}_\mathbf{p}$ *between two term vectors gives the correlation between the terms over the set of documents. The matrix product* $\mathbf{X}\mathbf{X}^\mathbf{T}$ *contains all these dot products. Element (i,p)(which is equal to element (p,i)) contains the dot product* $\mathbf{t}_\mathbf{i}^\mathbf{T}\mathbf{t}_\mathbf{p}$*. Likewise, the matrix* $\mathbf{X}^\mathbf{T}\mathbf{X}$ *contains the dot products between all the document vectors, giving their correlation over the terms* $\mathbf{d}_\mathbf{j}^\mathbf{T}\mathbf{d}_\mathbf{q} = \mathbf{d}_\mathbf{q}^\mathbf{T}\mathbf{d}^\mathbf{T}$.

Now, from **Theorem 1***, the singular value decomposition of an* ***m x n*** *real matrix* **X** *is a factorization of the form* $\mathbf{USV}^\mathbf{T}$*, where* **U** *and* **V** *are orthogonal matrices and* **S** *is a diagonal matrix with non-negative real numbers on the diagonal.*

The matrices giving us the term and document correlations become $\mathbf{XX^T} = \mathbf{USS^TU^T}$*and* $\mathbf{X^TX} = \mathbf{VS^TSV^T}$. $\mathbf{U}$ *contains the eigenvalues of* $\mathbf{XX^T}$*and* $\mathbf{B}$ *contains the eigen values of* $\mathbf{X^TX}$*.*

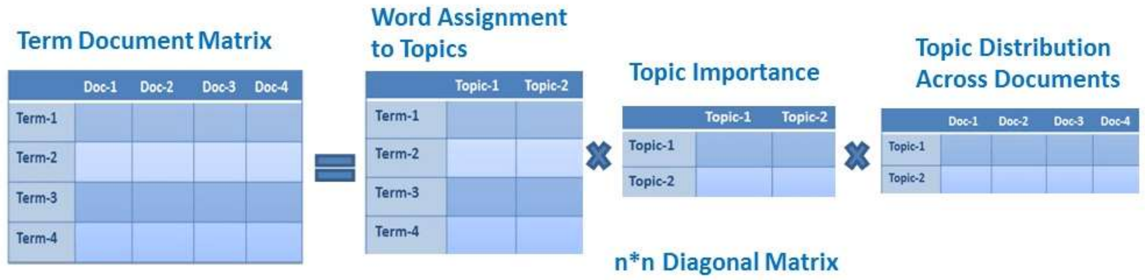
The decomposition of the matrices looks like this

$$\begin{array}{ccccccccc} & X & & & U & & \Sigma & & V^T \\ & (\mathbf{d}_j) & & & & & & & (\hat{\mathbf{d}}_j) \\ & \downarrow & & & & & & & \downarrow \\ (\mathbf{t}_i^T) \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix} & = & (\hat{\mathbf{t}}_i^T) \rightarrow & \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot & \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix} \end{array}$$

It turns out that when you select the $\mathbf{k}$ *largest singular values, and their corresponding singular vectors from* $\mathbf{U}$ *and* $\mathbf{V}$*, you get the rank* $\mathbf{k}$ *approximation to* $\mathbf{X}$ *with the smallest error.*

$$\mathbf{X_k} = \mathbf{U_k}\,\mathbf{S_k}\,\mathbf{V_k^T}$$

This approximation has a minimal error. But more importantly we can now treat the term and document vectors as a "semantic space". We can now compare document vectors and even term vectors from the respective matrices and gain insights from the result obtained from decomposition.



4.3 Implementation

We implemented LSA by first performing text pre-processing to remove unwanted lexicons and stop words to reduce noise and dimensions of our features. We then transformed our documents into a vector space model by converting them into *n*-dimensional TF-IDF vectors. We finally applied Singular Value Decomposition (SVD) on these vectors and then truncated them by picking the ***k*** largest singular values and then returned the components to get our term-topic and document-topic matrices. The weights of the topics with respect to the terms and the documents can be obtained from the results of the decomposition.

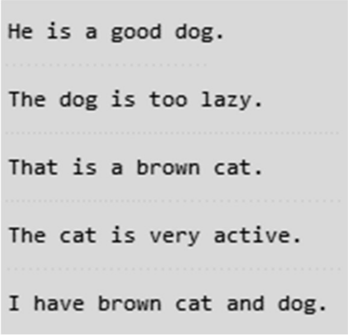
4.4 Algorithm

```
def LSA(k, X):

    X_clean = list(map(clean,X))

    vectorizer = TfidfVectorizer(smooth_idf=True)

    X_transformed = vectorizer.fit_transform(X_clean)

    svd = TruncatedSVD(n_components=k, algorithm='randomized', n_iter=300,
random_state=122)

    lsa = svd.fit_transform(X_transformed)

    column_names = ["Topic {}".format(str(i+1)) for i in range(k)]

    document_topic_matrix = pd.DataFrame(lsa,columns=column_names,index=X)

    dic = vectorizer.get_feature_names()

    term_topic_matrix = pd.DataFrame(svd.components_, index = column_names,
columns = (dic)).T

    return document_topic_matrix, term_topic_matrix
```

4.5 Final Implementation

We implemented Latent Semantic Analysis from scratch using Singular Value Decomposition on the vector space models. We initially started by performing pre-processing of our corpus.

Let us assume we have the following five sentences about cats and dogs.



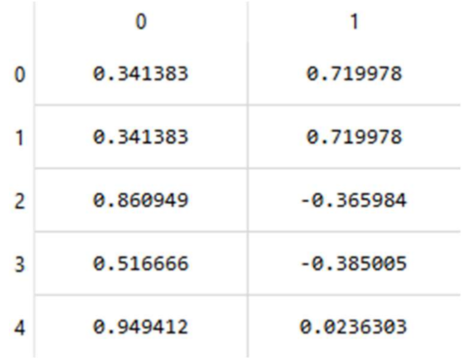
```
He is a good dog.

The dog is too lazy.

That is a brown cat.

The cat is very active.

I have brown cat and dog.
```

Documents 1, 2, 3 and 4 are about cats and dogs respectively

while document 5 is about both

These sentences represent two topics, namely cats and dogs. We would like to know the underlying topic distribution across these documents and which terms in each of these documents help us assign a topic to them.

After pre-processing and converting them into a vector space model, we performed truncated SVD with the number of components as 2 (since we have two topics) and obtained the low rank approximation matrix **X**.

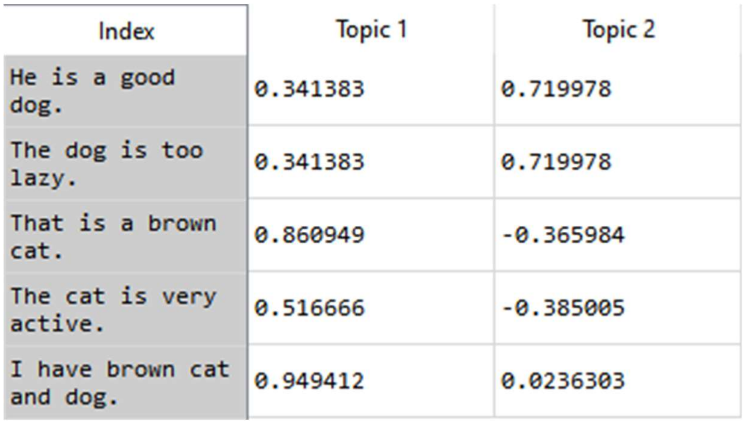


	0	1
0	0.341383	0.719978
1	0.341383	0.719978
2	0.860949	-0.365984
3	0.516666	-0.385005
4	0.949412	0.0236303

Low rank approximation matrix obtained

on performing SVD

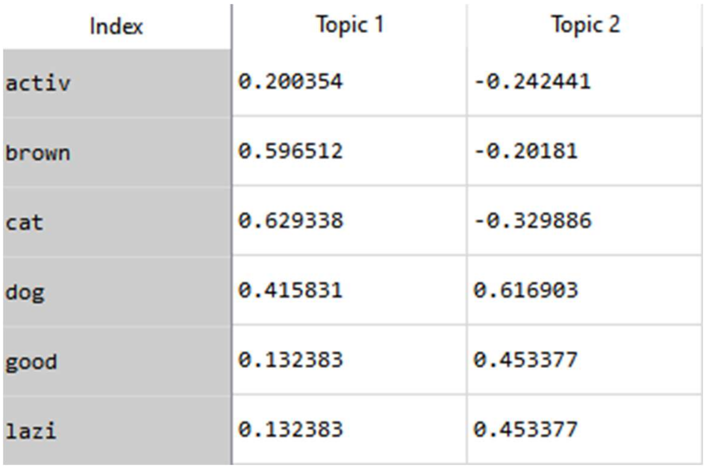
To analyse these scores in a more meaningful way, we reconstruct the document-topic and term-topic matrices using the components of the SVD. After constructing these, we obtain the following



Index	Topic 1	Topic 2
He is a good dog.	0.341383	0.719978
The dog is too lazy.	0.341383	0.719978
That is a brown cat.	0.860949	-0.365984
The cat is very active.	0.516666	-0.385005
I have brown cat and dog.	0.949412	0.0236303

Document-Topic Matrix

We can see a clear topic distinction between the documents. We can also conclude that Topic 1 is about cats and Topic 2 is about dogs and the first four documents have been appropriately scored. The Document-Topic matrix returns a score of how strongly a document aligns itself with the given topic and a higher score implies a strong correlation to the topic.



Index	Topic 1	Topic 2
activ	0.200354	-0.242441
brown	0.596512	-0.20181
cat	0.629338	-0.329886
dog	0.415831	0.616903
good	0.132383	0.453377
lazi	0.132383	0.453377

Term-Topic Matrix

We can see how each term in our corpus affects the topic distribution. As expected, terms such as “dog” and “lazy” are strongly related to topic 2, which is about dogs and terms such as “cat” and “brown” are strongly related to topic 1 which is about cats.

4.5 Conclusions

We can thus extract the underlying topics from any given set of documents by converting it into a vector space and then applying truncated SVD. The low rank approximation resultant matrix from the decomposition holds information regarding the topic-term scores and the topic-document scores. We can extract these components from the SVD to obtain our Term-Topic and Document-Topic matrices. The number of extracted topics is a hyperparameter (one which is provided to the algorithm and not learnt on its own) and this decides the truncating size and the dimensions of the two Topic matrices. It is often difficult to find the right number of topics to extract from a corpus, and clustering methods have proved successful in approximating the right number of topics for information extraction.

4.6 Improvements

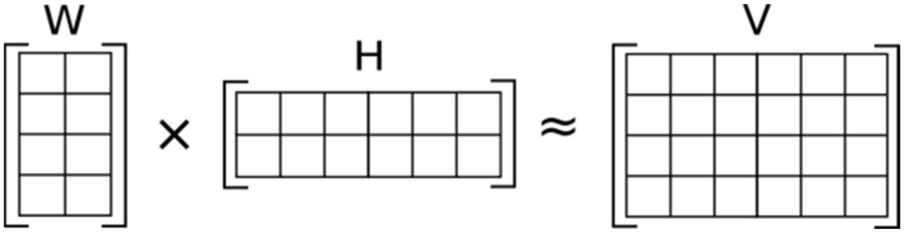
LSA is a highly optimised algorithm to perform topic modelling, but it does have a few drawbacks[4] with respect to information retrieval applications. The results of LSA are difficult to interpret, and it might not be feasible to construct Topic matrices for every decomposition. LSA also does not completely capture polysemy (multiple meanings of the same word) since each occurrence is assumed to have the same semantics. LSA can further be improved upon by using more advanced vector space models like Word2Vec or Doc2Vec which retain semantics as well as context. Modifying TF-IDF to use *n*-grams instead of individual words is also a popular method, along with the recently developed Probabilistic LSA[5] and LDA.

Currently, Non-Negative Matrix Factorization or NMF is preferred to perform topic modelling and segmentation since NMF allows for high readability and interpretation of data. It is also faster and easier to compute compared to LSA, since it factorizes the Term-Document matrix into only two sub-matrices.

5. Non-Negative Matrix Factorization - NMF

5.1 Linear Algebra Background

Theorem 1. *Non-negative matrix factorization (NMF or NNMF), also non-negative matrix approximation is a group of algorithms in multivariate analysis and linear algebra where a matrix* **V** *is factorized into two matrices* **W** *and* **H***, with the property that all three matrices have no negative elements.*



Theorem 2. *When* **V = WH***, the multiplicative factor of* **W** *and* **H** *is* **I**.

5.1 Mathematical Background

Non-negative Matrix Factorization is a Linear-algebraic model that factors high-dimensional vectors into a low-dimensionality representation. Like Principal component analysis (PCA), NMF takes advantage of the fact that the vectors are non-negative. By factoring them into the lower-dimensional form, NMF forces the coefficients to also be non-negative.

Let **V** *be a matrix where* $\mathbf{V}_{\mathbf{ij}}$ *describes the occurrence of term* **i** *in document* **j** *(this can be, for example, the frequency).* **V** *will look like this.*

$$\begin{array}{cc} & \begin{array}{ccccc} & & \mathbf{d}_j & & \\ & & \downarrow & & \end{array} \\ \mathbf{t}_i^T \rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{bmatrix} \end{array}$$

t$_i$ *stands for term* **i** *and* **d**$_j$ *stands for document* **j**

From **Theorem 1***, we know that this matrix can be factorized into two matrices* **W** *and* **H** *that is,* **V = WH***. There are several ways to obtain these matrices, the most conventional being Lee and Seung's multiplicative update rule which uses an iterative approach to find* **W** *and* **H** *until convergence.*

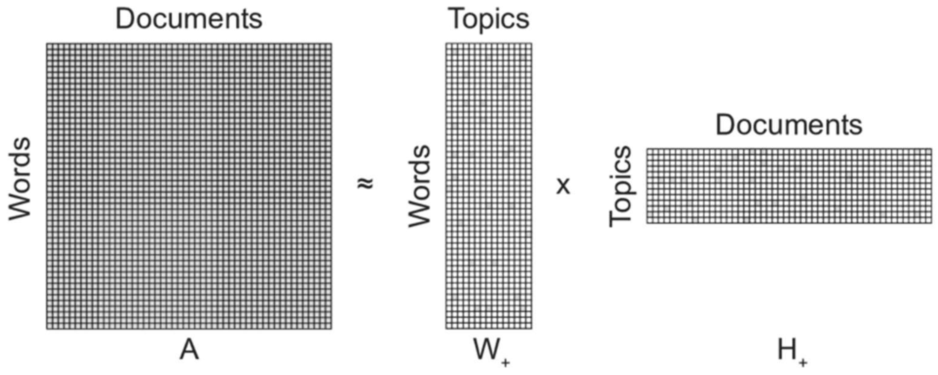
Initialize **W** *and* **H** *as non-negative matrices. We then update the values in* **W** *and* **H** *by computing the following, with* **n** *as an index of the iteration until* **W** *and* **H** *are stable.*

$$\mathbf{H}^{n+1}_{[i,j]} \leftarrow \mathbf{H}^{n}_{[i,j]} \frac{((\mathbf{W}^n)^T\mathbf{V})_{[i,j]}}{((\mathbf{W}^n)^T\mathbf{W}^n\mathbf{H}^n)_{[i,j]}}$$

$$\mathbf{W}^{n+1}_{[i,j]} \leftarrow \mathbf{W}^{n}_{[i,j]} \frac{(\mathbf{V}(\mathbf{H}^{n+1})^T)_{[i,j]}}{(\mathbf{W}^n\mathbf{H}^{n+1}(\mathbf{H}^{n+1})^T)_{[i,j]}}$$

When multiplying matrices **W** *and* **H***, the dimensions of the factor matrices may be significantly lower than those of the product matrix and it is this property that forms the basis of NMF. NMF generates factors with significantly reduced dimensions compared to the original matrix. For example, if* **V** *is an* **m** × **n** *matrix,* **W** *is an* **m** × **p** *matrix, and* **H** *is a* **p** × **n** *matrix then* **p** *can be significantly less than both* **m** *and* **n**.

The matrices **W** *and* **H** *hold information about the documents and its features, along with the underlying topics.* **W** *translates to a Term-Topic matrix which stores the topics associated with each term in our corpus and* **H** *translates to a Topic-Document matrix which tells us about the underlying topics in each document of our corpus. Since both* **W** *and* **H** *are non-negative matrices, the minimum score within these matrices is zero with the highest being one.*



5.3 Implementation

We implemented NMF by again performing text pre-processing to remove unwanted lexicons and stop words to reduce noise and dimensions of our features. We then transformed our documents into a vector space model by converting them into *n*-dimensional TF-IDF vectors with +1 IDF smoothening factor. We finally applied Non-Negative Matrix Factorization (NMF) on these vectors and obtained our term-topic and document-topic matrices which were returned. The weights of the topics with respect to the terms and the documents can be obtained from the results of the decomposition.

5.4 Algorithm to perform NMF

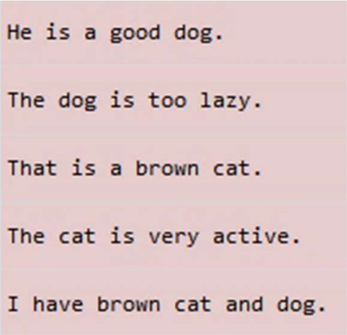
Performing NMF on the Term-Document matrix almost immediately gives us the two components. We finally put the components together along with the documents and terms and return the Term-Topic and Document-Topic matrices.

```
def NMFFunction(k, X):
    X_clean = list(map(clean,X))
    vectorizer = TfidfVectorizer(smooth_idf=True)
    X_transformed = vectorizer.fit_transform(X_clean)
    model = NMF(n_components=k, random_state=122, init='nndsvd')
    topicmodel = model.fit_transform(X_transformed)
    column_names = ["Topic {}".format(str(i+1)) for i in range(k)]
    document_topic_matrix =
pd.DataFrame(topicmodel,columns=column_names, index = X)
    dic = vectorizer.get_feature_names()
    term_topic_matrix = pd.DataFrame(model.components_, index =
column_names, columns = (dic)).T
    return topicmodel, document_topic_matrix, term_topic_matrix
```

5.5 Final Implementation

We implemented NMF on the vector space models. We initially started by performing pre-processing of our corpus.

Let us assume we are again working with the same example about about cats and dogs.

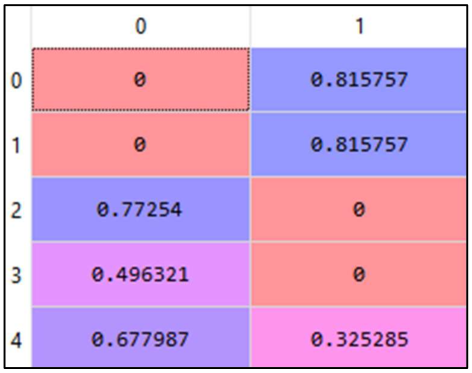


```
He is a good dog.

The dog is too lazy.

That is a brown cat.

The cat is very active.

I have brown cat and dog.
```

Documents 1, 2, 3 and 4 are about cats and dogs respectively

while document 5 is about both

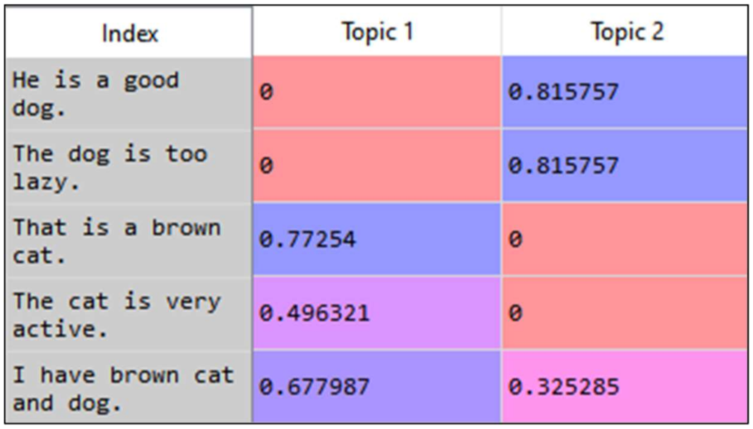
After pre-processing and converting them into a vector space model, we performed NMF with the number of components as 2 (since we have two topics) and obtained the product matrix **V**.



	0	1
0	0	0.815757
1	0	0.815757
2	0.77254	0
3	0.496321	0
4	0.677987	0.325285

Matrix **V** *obtained from* **W x H**

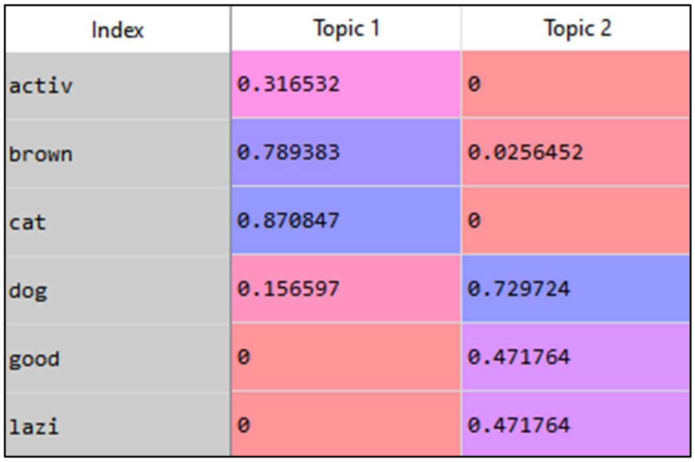
To analyse these scores in a more meaningful way, we reconstruct the document-topic and term-topic matrices as done before. After constructing these, we obtain the following matrices.



Index	Topic 1	Topic 2
He is a good dog.	0	0.815757
The dog is too lazy.	0	0.815757
That is a brown cat.	0.77254	0
The cat is very active.	0.496321	0
I have brown cat and dog.	0.677987	0.325285

Document-Topic Matrix

We can view the topic distribution between the documents. Like before we can also conclude that Topic 1 is about cats and Topic 2 is about dogs and the first four documents have been appropriately scored. However, the advantage here is that unlike LSA, NMF only returns non-negative values, so if a document does not align itself with a topic, its topic strength for that topic is zero. This allows for clearer interpretation and readability and can be used for topic classification purposes if we apply a suitable activation function.



Index	Topic 1	Topic 2
activ	0.316532	0
brown	0.789383	0.0256452
cat	0.870847	0
dog	0.156597	0.729724
good	0	0.471764
lazi	0	0.471764

Term-Topic Matrix

We can again similarly see how each term in our corpus affects the topic distribution. Just like before, since the minimum score returned is zero, it is easier to interpret these values and given any term in our corpus, we can easily classify it into one of the segmented topics.

5.5 Conclusions

We can thus extract the underlying topics from any given set of documents by converting it into a vector space and then applying Non-Negative matrix Factorization. The advantage with NMF is that it not only retains the semantic space, but it also provides better interpretation of the factorized matrix. However, unlike LSA which returns the singular values in matrix **S** which tells us about the importance of each topic, NMF does not give us this interepretation.

Since NMF returns only non-negative values, each value can be the strength or the confidence of a document belonging to a topic, with higher values indicating it is strongly related to that topic. Just like LSA, NMF also uses a hyperparameter **k** to segment the corpus into topics and choosing the right value of **k** is crucial to the algorithm’s accuracy. NMF tends to return a better segmentation of topics compared to LSA and is used in multiple applications not limited to natural language processing such as bioinformatics and search engine recommendations.

5.6 Improvements

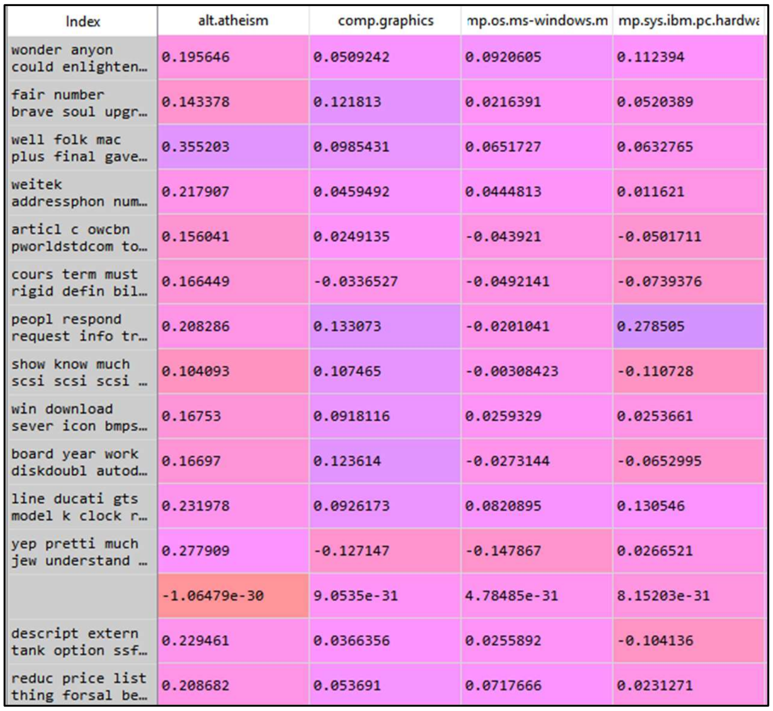
Although NMF is a powerful linear algebraic technique to perform matrix factorization, it does have its own downsides. NMF works better when the matrix is sparse since it the algorithm assumes missing values by default. LSA also follows a more deterministic approach instead. Modern developments have been made to improve the performance of NMF by evaluating and ranking the bases in the parallel multiplicative generation algorithm. NMF can further be improved upon by finding the right number of topics by performing a simple cluster analysis and then performing topic modelling to generate the exact segmentation of topics in our corpus.

6 Implementation

The input to our problem is the 20 News Group dataset, which contains about 20,000 news articles across 20 topics. We will be using both the linear algebraic approaches of topic modelling – LSA and NMF with the value of the hyperparameter **k** as 20 (since there are 20 categories). We will generate the Term-Topic and Document-Topic matrices as done before and compare them. We will then proceed to perform Uniform Manifold Approximation and Projection (UMAP), which will reduce our dimensions so we can view the document cluster on a 2D plot which will show us the distribution of topics in a two dimensional space.

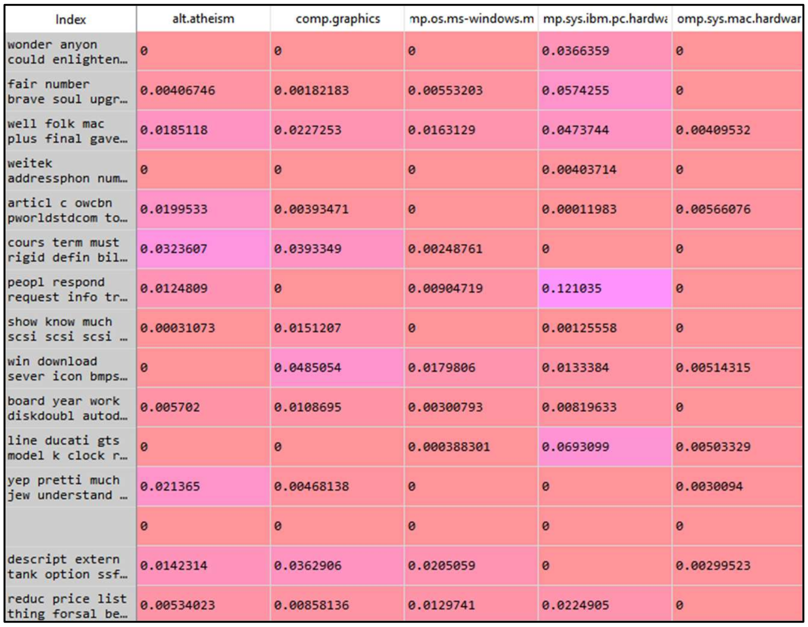
7 Conclusion

After loading the dataset and performing topic modelling, we obtain the following Term-Topic and Document-Topic matrices. Listed below are the top few entries in each of the Document-Topic matrices



Index	alt.atheism	comp.graphics	mp.os.ms-windows.m	mp.sys.ibm.pc.hardwa
wonder anyon could enlighten…	0.195646	0.0509242	0.0920605	0.112394
fair number brave soul upgr…	0.143378	0.121813	0.0216391	0.0520389
well folk mac plus final gave…	0.355203	0.0985431	0.0651727	0.0632765
weitek addressphon num…	0.217907	0.0459492	0.0444813	0.011621
articl c owcbn pworldstdcom to…	0.156041	0.0249135	-0.043921	-0.0501711
cours term must rigid defin bil…	0.166449	-0.0336527	-0.0492141	-0.0739376
peopl respond request info tr…	0.208286	0.133073	-0.0201041	0.278505
show know much scsi scsi scsi …	0.104093	0.107465	-0.00308423	-0.110728
win download sever icon bmps…	0.16753	0.0918116	0.0259329	0.0253661
board year work diskdoubl autod…	0.16697	0.123614	-0.0273144	-0.0652995
line ducati gts model k clock r…	0.231978	0.0926173	0.0820895	0.130546
yep pretti much jew understand …	0.277909	-0.127147	-0.147867	0.0266521
	-1.06479e-30	9.0535e-31	4.78485e-31	8.15203e-31
descript extern tank option ssf…	0.229461	0.0366356	0.0255892	-0.104136
reduc price list thing forsal be…	0.208682	0.053691	0.0717666	0.0231271

Document-Topic Matrix from Latent Semantic Analysis



Index	alt.atheism	comp.graphics	mp.os.ms-windows.m	mp.sys.ibm.pc.hardwa	omp.sys.mac.hardwar
wonder anyon could enlighten…	0	0	0	0.0366359	0
fair number brave soul upgr…	0.00406746	0.00182183	0.00553203	0.0574255	0
well folk mac plus final gave…	0.0185118	0.0227253	0.0163129	0.0473744	0.00409532
weitek addressphon num…	0	0	0	0.00403714	0
articl c owcbn pworldstdcom to…	0.0199533	0.00393471	0	0.00011983	0.00566076
cours term must rigid defin bil…	0.0323607	0.0393349	0.00248761	0	0
peopl respond request info tr…	0.0124809	0	0.00904719	0.121035	0
show know much scsi scsi scsi …	0.00031073	0.0151207	0	0.00125558	0
win download sever icon bmps…	0	0.0485054	0.0179806	0.0133384	0.00514315
board year work diskdoubl autod…	0.005702	0.0108695	0.00300793	0.00819633	0
line ducati gts model k clock r…	0	0	0.000388301	0.0693099	0.00503329
yep pretti much jew understand …	0.021365	0.00468138	0	0	0.0030094
	0	0	0	0	0
descript extern tank option ssf…	0.0142314	0.0362906	0.0205059	0	0.00299523
reduc price list thing forsal be…	0.00534023	0.00858136	0.0129741	0.0224905	0

Document-Topic Matrix from Non-Negative Matrix Factorization

We can hence see that the sparse matrix obtained by performing NMF is easier to interpret and understand compared to the matrix obtained by LSA since topics that are not related to a document are given the minimum score of zero.

We can further analyse our results by viewing both the Term-Topic matrices.

Topic	**Words**
alt.atheism	would use one get like know think 14eople
comp.graphics	window use thank file card drive email program
comp.os.ms-windows.misc	window file game team program run win play
misc.forsale	would like window get car think know problem
sci.space	file use like get pleas one 14eople look
talk.religion.misc	one get team window player would anyon good
rec.sport.hockey	would game card sale include offer new price

Term-Topic Matrix from Latent Semantic Analysis

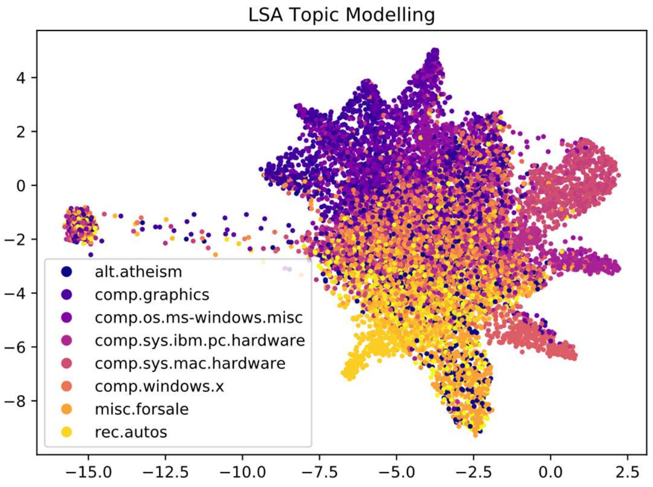
Topic	Words
alt.atheism	peopl us right say make law gun go
comp.graphics	use port help set program also printer imag
comp.os.ms-windows.misc	game team play player year win season last
misc.forsale	new sale price offer includ ship sell condit
sci.space	like look sound bike someth thing good realli
talk.religion.misc	armenian israel isra jew kill arab turkish attack
rec.sport.hockey	know anyon heard want game anyth wonder let

Term-Topic Matrix from Non-Negative Matrix Factorization

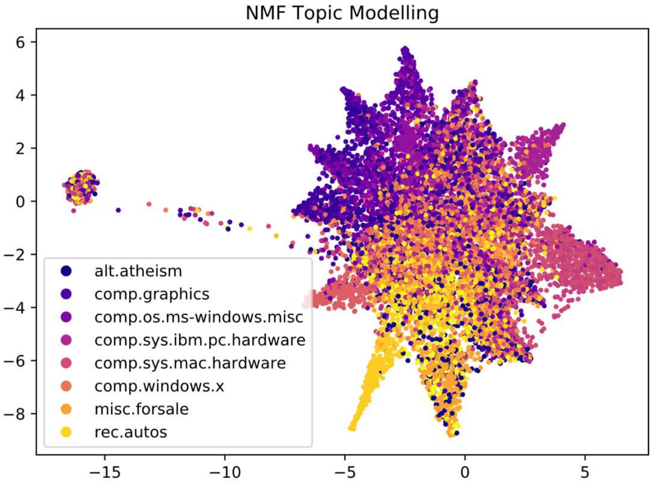
We observe from the above concept that the topics have a lot of overlapping words, causing a term to get segmented into multiple topics. However, we can also conclude that NMF performs better than LSA as shown from the relevant terms in each topic. We proceeded to plot our data points to find the reason behind our overlapping terms and to find out which algorithm performed better on the entire dataset.

7 Visualisations

We performed dimension reduction using Uniform Manifold Approximation and Projection (UMAP) on our low rank approximation matrix after performing LSA and our approximate product matrix after performing NMF. After reducing the dimensions of the Document-Topic matrix to a two-dimensional space, we plotted our results to find the topic segmentation across the documents.



We observe that there are a lot of overlaps between the topics and the terms, resulting in ineffective clustering of topics. Each colour represents a different topic and each point on the scatter plot is a document. We can also see that completely unrelated topics such as atheism and automobiles have zero overlaps but related topics such as Windows, hardware and IBM have a large overlapping section in the middle of the cluster. However, we can also observe that there are many data points which couldn't be classified into one topic alone and have terms which might contribute to multiple topics, as shown by the stray point in the left region of the plot.



We can observe from the NMF plot that the topics are better segregated and unrelated topics are quite well apart from each other. We can also observe that the boundaries are more well defined and all documents under the same topic are in close vicinity of each other. We can also conclude that NMF has done a better job at identifying the topics and the terms associated with each topic since there are lesser number of stray points which represent documents with multiple underlying topics.

8 Conclusion

We can thus find the underlying the topics in any document using simple yet powerful linear algebraic methods for topic modelling. Although NMF is faster and less computationally expensive to perform, it does not give us insights about how important the topics are and the weightage they carry in each document. However, LSA apart from being slower to compute, is also less interpretable compared to NMF and the matrices obtained after performing LSA are harder to read compared to the ones obtained after performing NMF since unlike NMF, LSA returns not only the terms related to a topic, but also the unrelated terms adding to clutter and disorganization. However, it is preferred over NMF since knowing the weightage each topic carries in a document is extremely significant and gives us more insights which portions of a document could hold valuable information. LSA is also used in other natural language processing applications such as text summarization and information extraction and retrieval.

9 Scope of Future Work

There have been a few improvements to the LSA algorithm over the years, the biggest being the introduction of p-LSA or Probabilistic LSA. Topic modelling has also been implemented using other nonlinear algebraic methods using LDA and Deep Learning techniques, which have provided state-of-the-art results in recent years. We can also perform simple EDA to find the optimum number of topics and feed that value into our LSA or NMF algorithm to further increase the accuracy of our topic segmentation.

10 Summary

- We used the 20 News Group Dataset to find the underlying topics and their relation to the documents and terms by removing stop words and unnecessary lexicons
- We converted these pre-processed documents into word vector models by converting them into TF-IDF vectors
- We then applied Latent Semantic Analysis and Non-Negative Matrix Factorization to extract the Term-Topic and Document-Topic matrices and compared them
- We finally plotted the topic segmentation in a two-dimensional space and analysed the topics

References

[1] https://en.wikipedia.org/wiki/Natural_language_processing

[2] https://en.wikipedia.org/wiki/Topic_model