CA683: Data Analytics and Data Mining

Name & Student Numbers	Javedali Shaikh - 16212373 Ken Brennock - 16213319 Sanjay Kumar Singh – 16211668 Kevin Shortall - 16213216
Programme	MCM
Module Code	CA683
Assignment Title	Unsupervised Text Categorisation
Submission date	17/04/2018
Module coordinator	Andrew McCarren

1. Introduction

Question: Can Unsupervised Machine Learning be used to categorize a corpus of text?

Doing text classification with supervised machine learning is not difficult and has state-of-the-art results in literature. But in the real world we don't always have 'target labels'. In fact, working in the software industry related to content, we can definitely say that we never have 'target labels' for massive data sets. So, we are left with the problem – unsupervised machine learning. Topic modelling, an unsupervised approach, is often used to perform clustering, but there are a lot of challenges when we want to do text classification. In our project, we did research and experimentation on this exact problem.

For the experimentation we have used '20 News Groups' [1] dataset, in which all documents are **pre-assigned** among 20 topics. Two different topic models (LDA and NMF) were implemented on the data, and the predicted topics received were compared with the original topics. For evaluation, confusion matrix was examined, and a novel approach (Dec 2017) known as 'Stability' was implemented.

Project-related files are stored at: https://github.com/javedsha/CA683-DataMiningAssignment

2. Proposed method

The CRISP-DM process model, which is a cross-industry standard process for data mining, was followed for this project. The following sections detail the steps taken through that process.

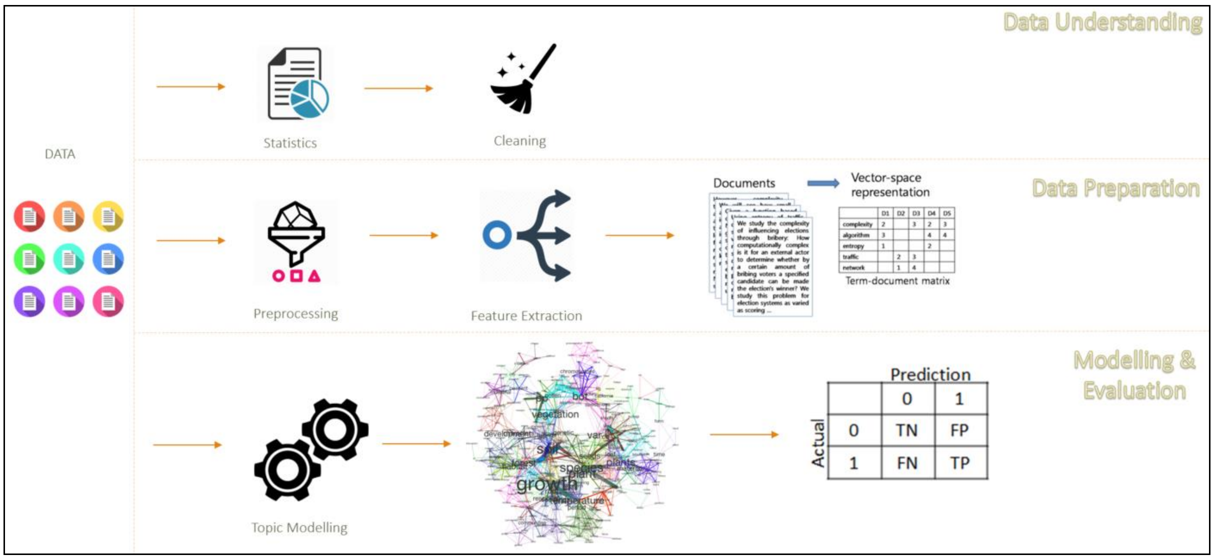


Fig 1: CRISP-DM process flow, tailored for our project

2.1 Data Understanding

The initial Data exploration gave some important insights into the dataset. There are approximately 18K documents in the 20 Newsgroup dataset of varying size, from 1 KB to approximately 44 KB. 20 Categories are already defined in the data. Each category had between 700 and 1200 documents. The wide range of document lengths we spotted initially, hinted that we might encounter a problem later during the modelling. The original dataset contains about 500K unique tokens (i.e. 'words').

It was clear from the topic names in the original dataset, that there are topics which are very closely related to each other. E.g. three topics related to 'Religion': *soc.religion.Christian; talk.religion.misc; alt.atheism.* two topics related to hardware: 'Mac.Hardware' and 'IBM.hardware'. We were already getting the feeling that this would cause problems during modelling phase.

***Note:** When doing topic modelling, we don't split the datasets into 'Train' and 'Test', like conventional machine learning approaches. Instead, we provide the entire datasets to the topic models (this is the standard procedure).*

2.2 Data Preparation

Many data cleaning and data preparation steps were taken to ensure the original data was correctly prepared for modelling. These included:

- Removing 'stop-words' – this was done using the library 'NLTK' written in python. These are words that occur so commonly in written English, that they are not useful in assigning topics to documents. Examples include: *'the', 'it', 'has'*.

  During the testing and investigation, it was found that, quite common words should also be removed which are not present in the 'stop-words'. For example, the word 'subject' was in nearly all the documents, therefore it does not add any information as to the topic of the document. Hence, a 'user defined' stop-words routine was also developed. Other examples for user defined words included, '.org', '.edu', '.com' etc.

- All punctuation was also removed.

- All numbers were also removed. This was done with the reasoning that, we didn't have any topic related to topics, and having numbers would impact the model performances. We tried a version with numbers, and we were seeing numbers present in the topics top K terms.

- Some non-English words were found when reading the documents. All non-English words were removed.

- Stemming was performed using Wordnet. This reduces the words to their root form. For example, the words *'fish', 'fished', 'fisher'* would all be reduced to the word *'fish'*.

- There were some recommendations on some sites that removing all unique words also helps with the modelling. As LDA is a probabilistic methodology intuitively this would make sense. Unique words do not add value when assigning a topic. As for NFM we used TF-IDF, which does the weighting, similar to what was done with LDA. *(Note: We don't use TF-IDF for LDA, as it has its own weighting scheme)*

All this data cleaning activity reduced the number of unique words ('tokens') to 25,174, the total corpus remained at about 300,000 words.

Also, we noticed that, without this cleaning, we were getting bad words from LDA/NMF. So, we were confident that these cleaning steps were improving the final performance of the models.

2.3 Modelling

The two models used in this project were Latent Dirichlet Allocation (LDA) [2] and Non-negative Matrix Factorization (NMF) [3]. Both models were implemented in Python. For LDA we used the library *'gensim'* [4] and for NMF we used another library *'scikit-learn'* [5].

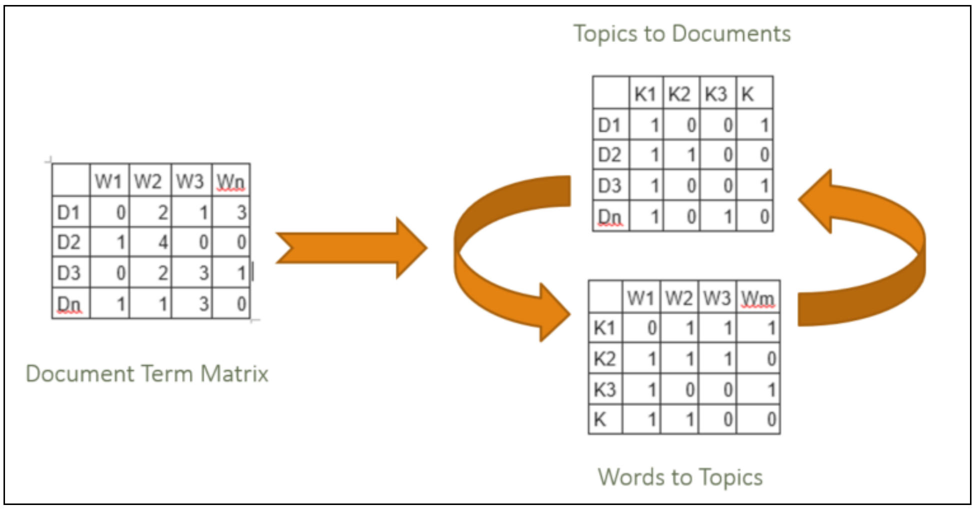
2.3.1 LDA (Latent Dirichlet Allocation)

The algorithm [13], [14] for LDA is relatively infinitive and is described below and shown in Fig 2.

1. Tell the algorithm how many topics. In this case, as there were already 20 topics, the number of topics was set to 20.
2. Assign a temporary topic to every word. This is not completely rando and it is done using a Dirichlet distribution (Anon., n.d.) (Sklar, 2017). This allocation of topics to word was not investigated in detail during the assignment.
3. The algorithm will check and update topic assignments based on the probabilities below. Looping through each word in every document and for each word its topics assignment is modified based on the two probabilities described below:
   I. Probability (prevalent) of the word across a topic – *p(topic t | document d)* = the proportion of words in document *d* that are currently assigned to topic *t*. This is the allocation of topics to this word across all documents.
   II. Probability (prevalent) that the topic is in the document - *p(word w | topic t)* = the proportion of assignments to topic *t* over all documents, that come from word *w*
4. Reassign the word with a new topic, where we pick the topic with the probability of:

*p(topic t | document d) * p(word w | topic t)*

All the data is allocated to the model at the very start and the documents are given a probability of belonging to a topic. It should be remembered that the topics are just a construct. In this case there were topics 0 to 19. The theory is that all the documents of the same topic will end up being in the same topic. It is then up to the human to give the topics names. In this assignment it was attempted to allocate the topics to the original news groups by reading the documents.



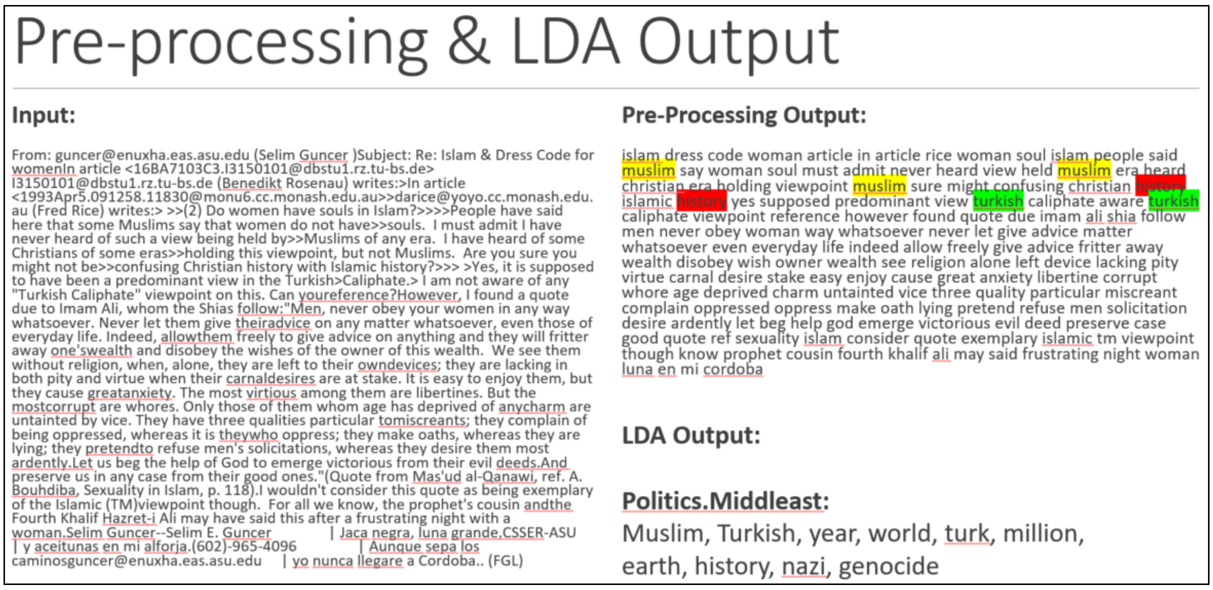
	W1	W2	W3	Wn
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
Dn	1	1	3	0

	K1	K2	K3	K
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
Dn	1	0	1	0

	W1	W2	W3	Wm
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K	1	1	0	0

Fig 2: LDA Matrix operations

The screenshot below shows the input, pre-processing output and LDA output.



Pre-processing & LDA Output

Input:

From: guncer@enuxha.eas.asu.edu (Selim Guncer)Subject: Re: Islam & Dress Code for womenIn article <16BA7103C3.I3150101@dbstu1.rz.tu-bs.de> I3150101@dbstu1.rz.tu-bs.de (Benedikt Rosenau) writes:>In article <1993Apr5.091258.11830@monu6.cc.monash.edu.au>>darice@yoyo.cc.monash.edu.au (Fred Rice) writes:> >>(2) Do women have souls in Islam?>>>>People have said here that some Muslims say that women do not have>>souls. I must admit I have never heard of such a view being held by>>Muslims of any era. I have heard of some Christians of some eras>>holding this viewpoint, but not Muslims. Are you sure you might not be>>confusing Christian history with Islamic history?>>> >Yes, it is supposed to have been a predominant view in the Turkish>Caliphate.> I am not aware of any "Turkish Caliphate" viewpoint on this. Can youreference?However, I found a quote due to Imam Ali, whom the Shias follow:"Men, never obey your women in any way whatsoever. Never let them give theiradvice on any matter whatsoever, even those of everyday life. Indeed, allowthem freely to give advice on anything and they will fritter away one'swealth and disobey the wishes of the owner of this wealth. We see them without religion, when, alone, they are left to their owndevices; they are lacking in both pity and virtue when their carnaldesires are at stake. It is easy to enjoy them, but they cause greatanxiety. The most virtious among them are libertines. But the mostcorrupt are whores. Only those of them whom age has deprived of anycharm are untainted by vice. They have three qualities particular tomiscreants; they complain of being oppressed, whereas it is theywho oppress; they make oaths, whereas they are lying; they pretendto refuse men's solicitations, whereas they desire them most ardently.Let us beg the help of God to emerge victorious from their evil deeds.And preserve us in any case from their good ones."(Quote from Mas'ud al-Qanawi, ref. A. Bouhdiba, Sexuality in Islam, p. 118).I wouldn't consider this quote as being exemplary of the Islamic (TM)viewpoint though. For all we know, the prophet's cousin andthe Fourth Khalif Hazret-i Ali may have said this after a frustrating night with a woman.Selim Guncer--Selim E. Guncer | Jaca negra, luna grande.CSSER-ASU | y aceitunas en mi alforja.(602)-965-4096 | Aunque sepa los caminosguncer@enuxha.eas.asu.edu | yo nunca llegare a Cordoba.. (FGL)

Pre-Processing Output:

islam dress code woman article in article rice woman soul islam people said muslim say woman soul must admit never heard view held muslim era heard christian era holding viewpoint muslim sure might confusing christian history islamic history yes supposed predominant view turkish caliphate aware turkish caliphate viewpoint reference however found quote due imam ali shia follow men never obey woman way whatsoever never let give advice matter whatsoever even everyday life indeed allow freely give advice fritter away wealth disobey wish owner wealth see religion alone left device lacking pity virtue carnal desire stake easy enjoy cause great anxiety libertine corrupt whore age deprived charm untainted vice three quality particular miscreant complain oppressed oppress make oath lying pretend refuse men solicitation desire ardently let beg help god emerge victorious evil deed preserve case good quote ref sexuality islam consider quote exemplary islamic tm viewpoint though know prophet cousin fourth khalif ali may said frustrating night woman luna en mi cordoba

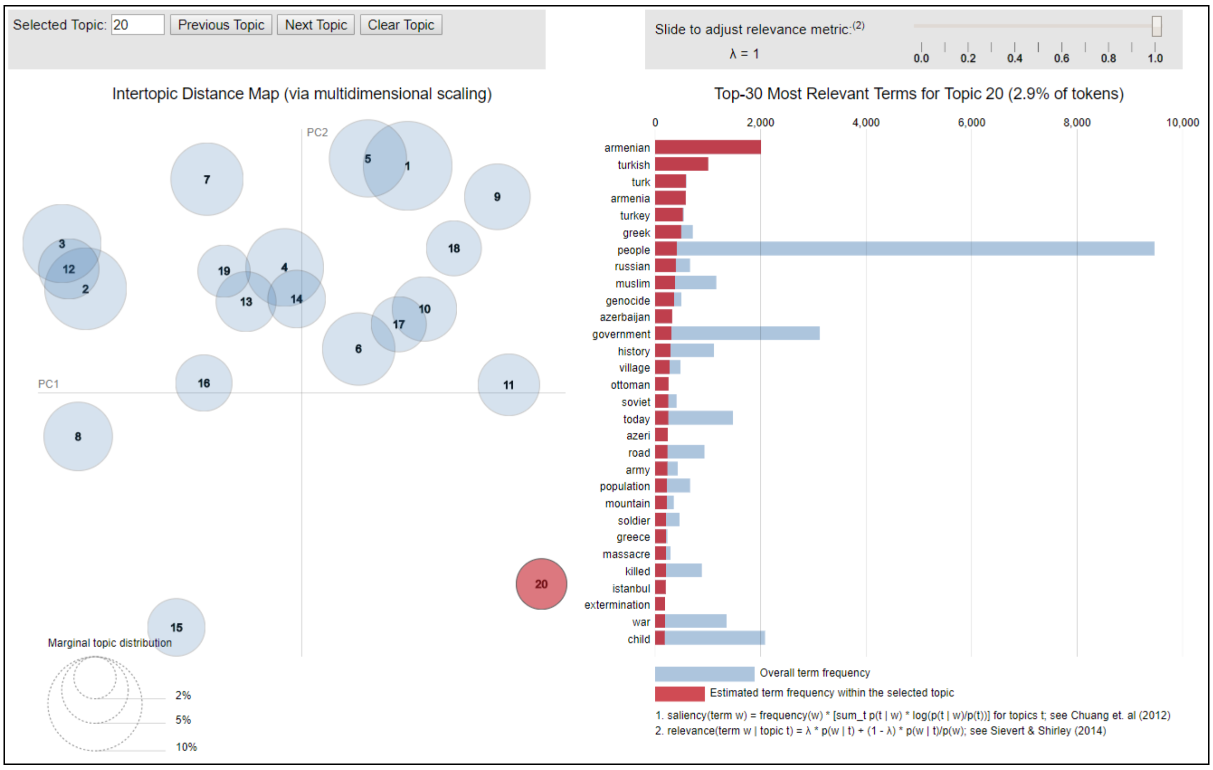
LDA Output:

Politics.Middleast:
Muslim, Turkish, year, world, turk, million, earth, history, nazi, genocide

Fig 3: LDA example output showing original (left) and cleaned/categorised (right) text in a document

2.3.1.1 Visualization

The LDA model can be visualised using the *LDAvis* package in R [6], which produces an interactive d3.js web interface, as shown in Fig 4 below. The reference code was sourced at [15].



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Fig 4: Interactive browser output from the *LDAvis* package in R

The interface helps to give clarity to LDA outputs by answering three main questions:

1. *How prevalent is each topic?* This is shown on the left side of the screen. Each topic is represented by a circle. The larger the circle, the more prevalent that topic across the corpus of documents.

2. *How do the topics relate to each other?* Again, this is shown on the left side of the screen. The distance between the circles is an approximation of the semantic relationship between the topics. The package uses Jensen-Shannon divergence to compute the inter-topic distances, and scaling is done through Principal Component Analysis.

3. *What is the meaning of each topic?* This is shown on the right by the 30 most relevant words of any topic which has been selected on the left. Each relevant word is represented by a horizontal bar chart, which can be made up of two colours. The first colour (red) shows how often that word occurs in just the selected topic, while the second colour (blue) shows how often that word occurs overall in the corpus. Thus, those words with a high proportion of blue bar chart are less effective at identifying this particular topic and are less relevant, as they occur very frequently elsewhere in the corpus. The lambda value can be adjusted to change the relevance of the displayed words: the lower the lambda value, the more relevant the displayed words are to just that particular topic.

2.3.2 NMF (Non-negative Matrix Factorisation)

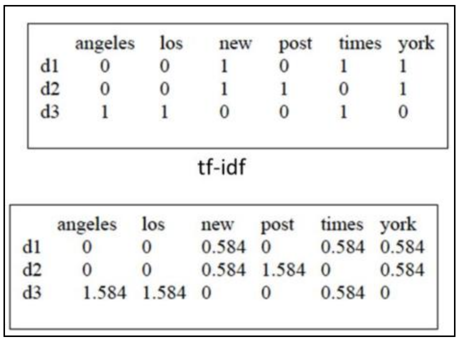
NMF [7] is a linear algebraic algorithm to factorize a matrix '*A*' into two smaller matrices, '*W*' and '*H*', with the property that all three matrices have no negative elements. This non-negativity makes the resulting matrices easier to inspect. This algorithm is widely used in dimensionality reduction techniques, clustering and topic modelling.

Before, we can use this algorithm we first have to convert our words into a vector representation (pre-processing and cleaning is already done). We used two approaches here:

1. **Bag of Words:** [8] Count the number of times words appears in a document and create a vector for each document. So, each document will be of the size of corpus vocabulary. So, in our corpus after cleaning we had ~25K unique tokens, so the size of each document vector will be 25K dimensions. Now, most of the entries in the vector will be 0 i.e. it will be a highly sparse matrix. Here, libraries use sparse matrices to reduce the memory consumption.

2. **TF-IDF:** [9] Bag of words is a good start, but it doesn't give any weights to the words. Words like *'high'* don't represent any topic in the dataset but could appear in all documents. Hence, such words should get less weighting compared to unique words like *'ball'* which highly indicates a sport topic. TF-IDF does the same thing: it gives less weighting to common words across the corpus, and more weighting to infrequent words across corpus.

The screenshot below in Fig. 5 represents the output of bag-of-words and TF-IDF.



	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

tf-idf

	angeles	los	new	post	times	york
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

Fig 5: TF-IDF example

Once we have our weighted vectors, we can feed them into the NMF model. Intuitively, NMF will factorize the input matrix *'A'* (i.e. the TF-IDF matrix shown above in Fig 5) and decompose this matrix into two matrices, *'W'* and *'H'*.

Matrix 'W' represents the document-topic relationship and matrix 'H' represents the word-topic relationship. As shown in Fig 6 below, in matrix '*W*' we can see that document 1 belongs to Topic 2, and document 2 belongs to topic 3. Similarly, in matrix '*H*', we can see that the word '*bank*' belongs to topic 3 and the word '*hospital*' belongs to topic 2.

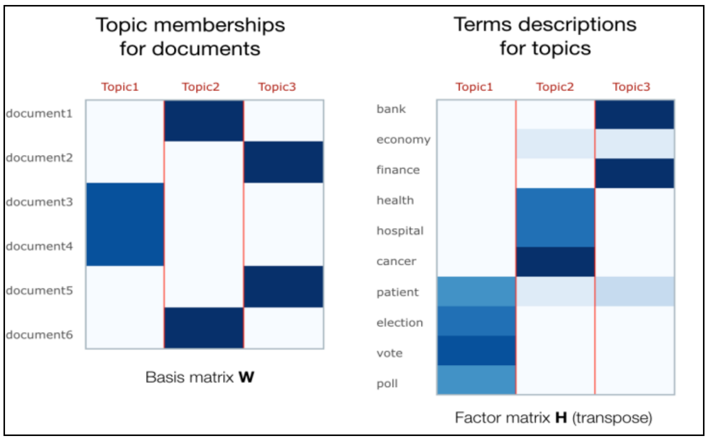
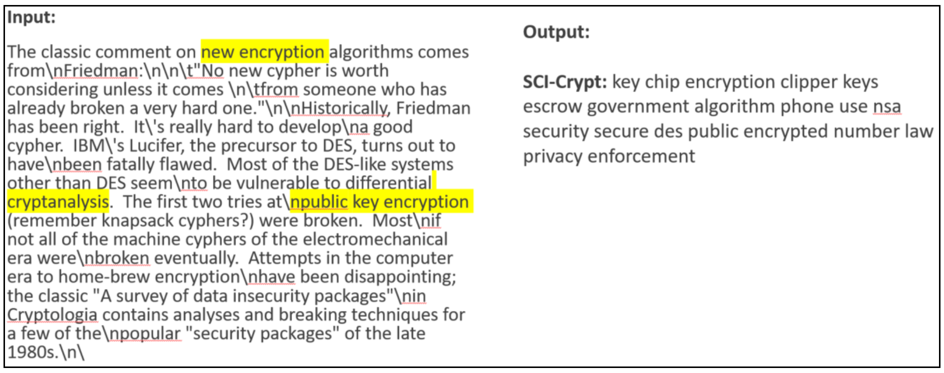


Fig 6: NMF explanatory charts

The screenshot below in Fig 7 shows an example input document and its NMF output. It successfully classified the document as being in the '*cryptography*' topic.



Input:

The classic comment on new encryption algorithms comes from\nFriedman:\n\n\t"No new cypher is worth considering unless it comes \n\tfrom someone who has already broken a very hard one."\n\nHistorically, Friedman has been right. It\'s really hard to develop\na good cypher. IBM\'s Lucifer, the precursor to DES, turns out to have\nbeen fatally flawed. Most of the DES-like systems other than DES seem\nto be vulnerable to differential cryptanalysis. The first two tries at\npublic key encryption (remember knapsack cyphers?) were broken. Most\nif not all of the machine cyphers of the electromechanical era were\nbroken eventually. Attempts in the computer era to home-brew encryption\nhave been disappointing; the classic "A survey of data insecurity packages"\nin Cryptologia contains analyses and breaking techniques for a few of the\npopular "security packages" of the late 1980s.\n\

Output:

SCI-Crypt: key chip encryption clipper keys escrow government algorithm phone use nsa security secure des public encrypted number law privacy enforcement

Fig 7: NMF example output showing original (left) and document topic 'sci-crypt' (right).

2.4 Evaluation

Two measures of evaluation were used in this project: Confusion Matrix and Stability.

2.4.1 Confusion Matrix:

We first manually labelled all topics from LDA and NMF, as they are abstract (Topic 0, 1, 2, etc.) This is a difficult, labour-intensive activity that does not scale well to larger topics. But we can do it for a small number of topics.

LDA/NMF will give 20 topics for each document, with topic probability for each document. We took the topic which had maximum probability.

The overall precision for all topics was very low for both models. But when we drilled down to each individual topic, we found good precision for some topics, some topics were having average precision (algorithm was confused as they were closely related), and some topics were missed (not enough data provided). Table 1 shows a sample of these results.

	NMF	**LDA**
Overall	**30%**	**32%**
Good Labels	• Crypt: 80% • Politics Middle-East: 68%	• Windows.X: 89% • Hockey: 86% • Politics.middle-east: 80%
Confused Labels (due to close relation)	• MAC.Hardware: 26% • IBM.Hardware: 42%	• Graphics (18%) • MAC.Hardware: (17%)
Missed Labels	• Religion Atheism • Medical	• Religion Atheism • Medical

Table 1: Confusion Matrix outcomes

Below is example of some mis-classified documents:

Document	Actual Topic	Predicted Topic
1	Mac.Hardware	IBM.Hardware
2	Comp.Windows	Comp.Windows.Misc
3	Autos	Motorcyles

Table 2: Some mis-classified documents

The reason for most of the mis-classification was assigning a document to a topic which has other closely related topics. E.g. as shown above, document 1 was misclassified as *IBM.Hardware*, whereas its actual topic is *Mac.Hardware*. This is because both topics are very closely related to each other, i.e. both topics share many common words.

Why were the documents allocated to the 'wrong' topic?

There are a variety of reasons that could be found: 1) the document just contained some of these words but with multiple instances, hence, pushing up the probability of the document belonging to this topic; 2) the documents were very small so even one word had a big impact on the topic allocation.

When investigating the results, the team found a couple of points that were suspected to impede a more positive outcome:

1. The documents are divided into news groups. However, even when reading some of these documents, it is difficult to see how they were put into a specific news group. Some of the news groups are similar in the types of topic, for example, as mentioned earlier, there are three topics on religion.
2. Before cleaning there was an enormous range in the size (number of words) of the documents. When pre-processed as described above some of the documents were left empty. Hence, they would be dropped from the model. However, there was still a large variation in the size of documents.
3. The documents are allocated to a topic based on the highest score they received. However, in some cases the scores could be close, and all below 50%.

2.4.2 Stability

This concept is take from a recent paper (Dec 2017) [10] written by Mark Belford from UCD, Insights Centre. Before this paper, there weren't any good evaluation metrics available for Topic Modelling. The main reason for that was that it is an unsupervised algorithm, i.e. no target labels.

We met Mark in an NLP meet-up in Zolando, and he happily shared the draft paper and the Git Hub code. *Note: We spent a good amount of time in understanding the original code from authors [11], changing it for our needs (different format and logical flow).*

Average Term Stability (ATS) – compares the similarity between two topic models based on a pairwise matching process at a topic level.

Uses the Jaccard Index:

$$Jac(R_i, R_j) = \frac{|R_i \cap R_j|}{|R_i \cup R_j|}$$

To carry out the experiment, we performed the below steps:

1. K = top 10 terms
2. Ran 10 'runs' of LDA (L1, L2, …, L10)
3. Output top k terms for 20 topics
4. Calculate ATS for each combination of (L1, L2, …., L10)
5. Take the mean ATS
6. Repeat steps 1-5 for k = 20, 30, 40 and 50
7. Repeat steps 1-6 for NMF

NMF: Stability increased as we increased the number of top terms per topics.
LDA: Stability was constant at 1. This is very strange and should not happen. But the reason for this behaviour could be small size of the dataset (after intensive cleaning).
Fig 8 below represents these findings:

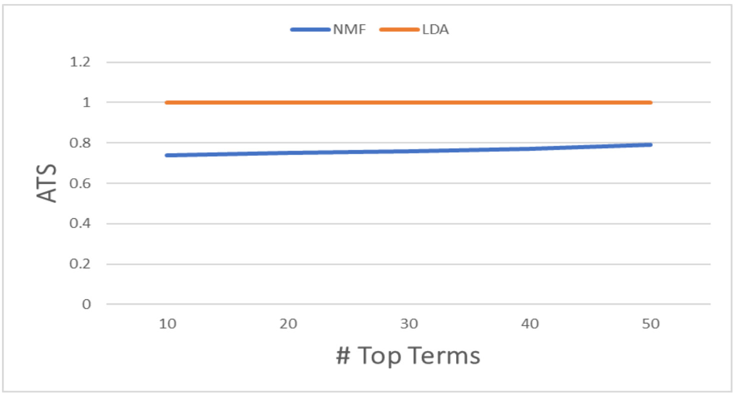


Fig 8: Term stability graph

3. Conclusion

1. Both models faced difficulties in identifying topics which are closely related (e.g. autos-bikes).

The solution to this, as per our research, is to have more data specially for topics which are similar. Also, to use advanced algorithms like 'Guided LDA' [12]. We did try to implement this algorithm for our project but due to time limits, we were not able to complete. The main workings of this algorithm are that you can provide seed topics/words with higher probability distribution (rather than uniform as in the case of original LDA). Then the algorithm successfully separates the closely related topics, even without the need of having extra data.

2. Varying sizes of documents impact the performance.

After our cleaning process, we found that there was a large mis-proportion in the number of words per document. Some documents had more words, but most documents had fewer. This has impacted the output of LDA & NMF.

3. More data would be required to improve the performance of models.

The general rule of thumb for Topic modelling algorithms is: the more data, the better it is, as it can cover a wider vocabulary range. But there is a relation between the number of topics we want and the data available. For example, in our case we wanted 20 topics and the size of unique tokens after cleaning was only 25,000. Immediately this looks reduced. On top of that we have other problems mentioned in point 1 and 2. To prove this reasoning, when we run the algorithms with fewer topics (7-10) and with the same data, we were getting higher precision for these 7-10 topics.

4. NMF runs much faster than LDA!

NMF is a linear algebraic process. And that says everything. It takes around 1 minute to run NMF, whereas LDA takes 20 minutes with the same settings.

5. Quality of words were better in LDA.

Upon manual evaluation, we found that LDA was outputting topics with better terms, i.e. the terms were more closely related to the topic.

6. Doing text classification is <u>hardest</u> with unsupervised algorithms!

Our final conclusion is: you can do text classification using unsupervised algorithms, **but it has to be on a smaller scale.** On a bigger scale it becomes really hard to manually assign topics from algorithms to a human name. For 20 topics, it took us ~2 hours. In real world, we could end up with 200-500 topics.

Our proposal for a real word project:

Create a sample dataset which represents your larger dataset. This dataset should contain categories/topics which are "high level", such as sports, news, finance, entertainment, autos, weather, lifestyle, etc. Also, this dataset should have target labels. For this, you can use domain experts (crowd sourcing) or internal resources, to label each document with one or more topic. This will give you your supervised data set (i.e. containing target labels).

Now you can run the entire experiment which is describe in this report, i.e. run unsupervised algorithms like LDA/NMF on sample data and let them create topics for you. Next, manually go through all the topics (not many) and give them a human name. After this, for each document in the sample dataset, obtain a 'single/multiple' topic using trained LDA/NMF. Once all your documents are assigned topic(s), you can compute the confusion matrix (precision/recall) and stability. Now two things can happen:

1. Numbers are satisfactory: You can now use this version of LDA/NMF (parameters, hyperparameters) and train it on a larger dataset.
2. Numbers are not satisfactory: Repeat the experiment with varying the parameters/hyperparameters, updates in model, doing ensemble, or updating your sample dataset, until you are happy with the results.

4. References

[1] - Ken Lang, http://qwone.com/~jason/20Newsgroups/

[2] – David M. Blei, Andrew NG and Michael I. Jordan, https://ai.stanford.edu/~ang/papers/nips01-lda.pdf

[3] – Daniel D. Lee and H.Sebastian Seung, https://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[4] - https://radimrehurek.com/gensim/

[5] - http://scikit-learn.org/

[6] - https://github.com/cpsievert/LDAvis

[7] - https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

[8] - https://en.wikipedia.org/wiki/Bag-of-words_model

[9] - https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[10] - Mark Belford, Brian Mac Namee, Derek Greene, Stability of Topic Modelling via Matrix Factorization, https://arxiv.org/abs/1702.07186

[11] - https://github.com/derekgreene/topic-stability

[12] - https://github.com/vi3k6i5/GuidedLDA

[13] - https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

[14] - https://www.quora.com/What-is-a-good-explanation-of-Latent-Dirichlet-Allocation

[15] - https://www.kaggle.com/solution/lda-visualization/code