Dissertation

Ryan Pollard

03/08/2021

Contents

Abstract

1. Text Analysis and NLP: A Background

Data is becoming increasingly more important and available to businesses and individuals alike. Businesses are generating and storing so much data and as a result there is a desire to investigate if this data could be of use. For example, a business might sell their goods on Amazon and receive thousands of customer reviews a day. Data is traditionally thought of as being spreadsheets with cells being populated by numbers, however, the textual-review data, as used as an example above, also has its uses. In the case of reviews,it represents an sign of the quality of that product. This "sign" is in essence a piece of data. In the same way, social media represents a vessel of data that represents the thoughts and feelings of it's users; that is, there are reams of textual data that indicate a feeling and opinion about a given topic. From casually browsing Twitter for example, a few tweets can be read regarding the latest news event and after reading five to ten tweets it could be concludde how the users of twitter are feeling about this event.

Twitter is one of the most popular social media platforms and plays a huge role in influencing and expressing public opinion. About 500 million tweets are published every day - where each tweet is a message limited to 288 characters. It's an important tool to understand peoples' emotion - this is done via "Sentiment analysis" of the tweets. It's also useful in discovering the most discussed topics and this is explored using statistical techniques known as "Topic Modeling".

That act of browsing and reading of tweets is effectively a data gathering exercise to discern the opinion of the people. Since language is the way humans interface and naturally share data, it's completely natural to humans to read these words on a screen, coming from multiple tweets, and form a single opinion about how people must be feeling about that topic. For example, looking at tweets that speak about tax increases, ten tweets may be read, seven of which may criticise this change in policy, two of which may praise it, and one that is in between those two feelings. From this a reader of these tweets can conclude that the attitude to increasing taxes is largely negative, with seven of the ten tweets carrying negative connotations.

In addition to tweets being useful in gauging opinion of it's users, as time goes on more and more people are using twitter - meaning there can be varied opinions on a particular topic. With the advent of smartphones across the globe and the ease of access of internet, the volume users voluntarily inputting text data has exploded; adding to the importance of being able to analyse this data in a statistically structured way.

In terms of how this data is analysed, the problem with text data is that it is unstructured - unlike numbers that represent financials or demographics with set numerical rules; and therefore it is inherently difficult to work with. Text often is hard to interpret to a machine, with it's grammar, syntax rules and complex patterns and as a result it is not straight forward to carry out statistical techniques on text data.

There are however rigorous statistical techniques that have been developed that allows text data to be processed statistically. Under the umbrella known as "NLP" - Natural Language Processing, these techniques aim to understand text-based data. Natural Language Processing, where a Natural Language is any language that evolved with humans to communicate, uses the linguistic rules that every language has in order to extract information from pieces of text. It enables computers to process and understand human language.

1.1 How computers use text data

First a definition, a "document" is a single separate piece of text. In the context of tweets, a document would be one tweet. If news articles were to be analysed, a document would be one article. Documents comprise a "corpus", which is all the documents combined.

Machine learning techniques traditionally need numerical data. A logistic regression is a simple machine learning tool that can express the segmentation of a dependent variable with two groups as a function of its independent variables. It's possible to have exactly the same framework using text data

In this toy example there ten documents with each document being a single tweet, five of which give misinformation about Covid-19, and five of which give true information about Covid-19. The aim is to build

a logistic regression such that the relationship between the text in each tweet and its label of misinformation/information is obtained. However,a logistic regression cannot do this because it can only interpret numerical data. The logical step then is to convert these documents (tweets, in the example) into numbers. This is done via "Vectorisation" of each document. The corpus of documents are then represented as a matrix where each row a document, having a vector of numbers and the columns are all the words in the corpus. The numbers in the matrix represent the amount of times a word in a column appears in a document (which is a row). Each row then represents all the text in one of the documents. This process converts text data into numerical data and as a result machine learning techniques can be used on this matrix.

Creating the matrix above requires certain steps so that it is primed for statistical analysis to be carried out. The process of splitting each document into it's individual words is called "Tokenisation". This is a list of words that appear in each document. For each document, this list of words is vectorised using one of the following techniques:

1. Bag of Words - a matrix where each word is a column, each row is a document (one piece of text data) and the number in each cell is the number of times the column's word appears in the document's (row) text. See the image below for an example.

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Figure 1: A toy example of a bag of words matrix

2. TF-IDF - stands for "Term Frequency — Inverse Document Frequency". This generates a matrix that assigns a weight to each word which signifies the importance of the word in the document and corpus. This is more sophisticated than the Bag of Words approach. The importance of a word in each document is based on the amount of times it appears in the document and also how rare the word is in the corpus. Rare words in document are given a higher weighting as rare words help distinguish what topic the document belongs to. For example, if a document contains a rare word "Nuclear", this is likely to be a good indicator that the document is about the topic "Energy". If a word appears frequently in a document but infrequently across the whole corpus, it is given a large weight.
3. word2vec - uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. This is even more sophisticated than the above as context of the word is taken into account.

Using one of these methods to convert a corpus of documents into a matrix, machine learning techniques can be used in order to classify documents and other operations.

1.2 Applications

- **Machine Translation** - translating text from one language to another. This is becoming more accurate with the aid of Deep Learning.
- **Speech Recognition** - Recognition of vocal language and converting it to text.
- **Question Answering Systems** - For example chatbots often used to interface between clientes and businesses. Another example is Personal Assistants such as Siri

- **Contextual Recognition** - Obtaining meaning of words from the context of the sentence, not just using the definition of the word.
- **Text Summarisation** - Taking a large piece of text and condensing it but retaining the original meaning .
- **Text Categorisation** - Classifying texts. For example, perhaps given a piece of text, a tweet in this example, analysing the words gives rise to ability to classify it as a tweet that contains misinformation about Covid-19 to a certain level of accuracy using machine learning techniques.
- **Text Analytics** - Deriving insights from text data. Methods include clustering, summarisation, sentiment analysis. An example application of this is Spam detection. Statistical techniques are used to classify certain emails as spam so they don't reach a user's inbox.
- **Topic Modelling** - Classifying a corpus of documents into topics.

1.3 Supervised and Unsupervised Machine Learning

Supervised machine learning is taking labeled data and using it to express the relationship between the labels and the independent variables in the data. In the running example, this would mean taking twitter data that express an opinion about Covid-19 and manually labeling them "misinformation" if it contains misinformation and "information" is it contains real information. The relationship could then be explored between the text and the labels and this relationship can be used to predict if a new, unlabelled and unread tweet as having misinformation to a certain percent of accuracy.

However, since there is exponentially more text information available, it is extremely inconvenient for these texts to be labeled since it requires a human to read each text. Therefore it is common to apply unsupervised (unlabeled) statistical techniques to this type of text data. There are a family of machine learning algorithms that try to discover latent hidden structures and patterns in unlabeled text data from their various attributes and features. Several unsupervised learning algorithms are also used to reduce the feature space, which is often reduced data from a higher dimension to one with a lower dimension. This dimensionality reduction can be seen as taking a large number of documents and assigning them into a relatively small amount of groups, where each group has its own topic which describes broadly the type of tweets the group contains.

2. Topic Models

2.1 Introduction

To serve as an introduction to Topic Model a toy example is considered: Data comprising twenty thousand tweets expressing opinions about Covid-19 is availableand using unsupervised machine learning techniques and the vectorisation of text explained above, the data can be mathematically expressed and analysed. These tweets can be grouped into three groups using machine learning algorithms with the following aim: tweets in group one contains text that refer to Covid-19 cures, group two contains tweets about Covid-19 vaccines and group three contain tweets about Covid-19 prevention. These groups can be interpreted to be topics. To reach this goal, a range of statistical techniques are used called Topic Models.

Topic models extract the distinguishing concepts, or topics, from the the corpus (that is, all the documents). It groups up documents into topics without human intervention and judgment. These topics can include opinions or facts. The models then use statistical techniques to explore the hidden and latent structures in the corpus in order to group up documents, where the hidden structures in the corpus refer to topics.

2.2 Methods for topic modeling

There are various algorithms to carry out Topic Modeling.

The following three methods are explored:

- Latent Semantic Indexing
- Latent Dirichlet Allocation
- Non-negative matrix factorization

2.3 Latent Semantic Indexing (LSA)

Latent Semantic Analysis takes the matrix that contains the documents and terms and decomposes it into a document-topic matrix and topic-term matrix - this is the TF-IDF matrix explained in Section 1.1. This matrix will be sparse since it contains all th terms in the corpus and a lot of terms will not be important in defining the topics. In order to find only a few topics this matrix needs to undergo dimensionality reduction. This reduction can be done using truncated Singular Value Decomposition (SVD). THis is a technique that factorises a matrix into the product of 3 separate matrices, as seen in 2. V represents the TF-IDF matrix. The matrix Σ represents the singular values of V and to keep the most important topics only the top t largest singular values are kept, which means keeping only the first t columns of U and $V*$. In other words, t is a hyperparameter which adjusts the number of topics outputted by the SVD.

U represents the document-topic matrix, $V*$ represents the term-topic matrix. For both matrices the columns represent topics. For U, the rows represent the document vectores expressed in terms of topics. The rows in V represents the term vectors expressed in terms of topics. The numbers in the cells represent how important the topic is for the term or document in $v*$ and U respectively. In this way, the documents can be assigned a topic by choosing the highest topic importance value.

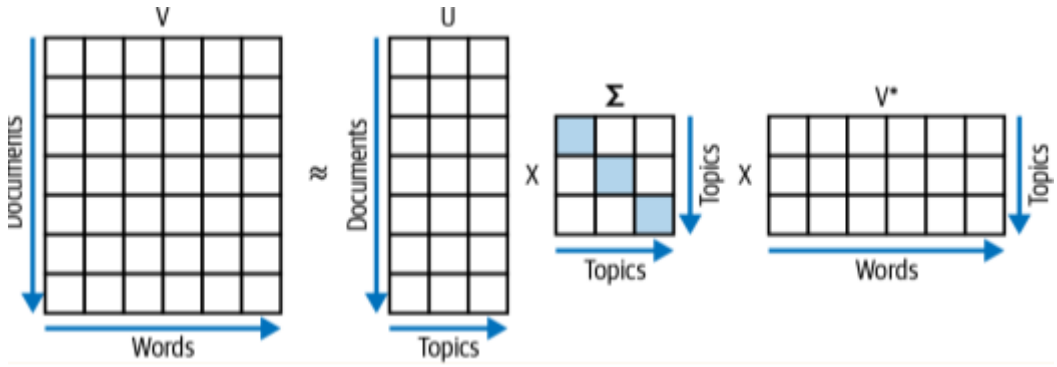


Figure 2: Singular Value Decomposition of the TF

2.4 Non-negative matrix factorisation

One way to find the hidden topics of a corpus is the factorization of the "document-term matrix" - this is the a matrix with the rows being the documents and the columns being the words in the whole corpus - and if a docyment contains a particular word in the corpus it gets a value of 1 in that cell where the column is that particular word. This matrix has only positive-value elements and so methods from linear algebra can be used to order to represent the matrix as the product of two other nonnegative matrices. The original matrix is called V, and the factors are W and H:

$$V \approx W \cdot H \tag{1}$$

In the context of text analytics, both W and H have an interpretation. The matrix W has the same number of rows as the document-term matrix and therefore maps documents to topics (document-topic matrix). H has the same number of columns as features, so it shows how the topics are constituted of features (topic-feature matrix). The number of topics (the columns of W and the rows of H) can be chosen arbitrarily. The smaller this number, the less exact the factorization.

The above two methods are computationally much cheaper than LDA but lacks in the ability to find multiple topics in single documents - this ability is gained by adding a Dirichlet prior on top of the data generating process, as discussed below.

3. Latent Dirichlet Allocation

3.1 LDA Introduction

LDA considers each document as consisting of a mixture of different topics and these topics are made up of a mixture of words. To ensure that the number of topics per document is low and to have only a few important words constituting the topics, LDA uses a Dirichlet prior in its statistical process. This is applied both for assigning topics to documents and for assigning words to the the topics. After these initial assignments, a generative process begins. It uses the Dirichlet distributions for topics and words and tries to re-create the words from the original documents with stochastic sampling. This process has to be iterated many times and is therefore computationally intensive. The results can be used to generate documents for any identified topic. For a given corpus of documents, each document can be represented as a statistical distribution of a fixed set of topics. LDA assumes that each document is generated by a statistical generative process governed by a document's topic distribution, and the topics' word distributions. After this generative process has taken place and when the documents generated that most closely resemble the actual documents in the corpus, the topic distribution of the documents can be observed to assign documents to topics.

3.2 LDA In-Depth

Topic models are based on the idea that each document, that is each piece of text, are a mixture of topics. For example, perhaps there is a news article about electric vehicles and how it impact the economy. In this example it can be said this document (i.e the article) is a combination different topics, not just one - it could well be 50% "Transport", 30% "Technology" and 20% "Economy". This distribution is labelled as θ.

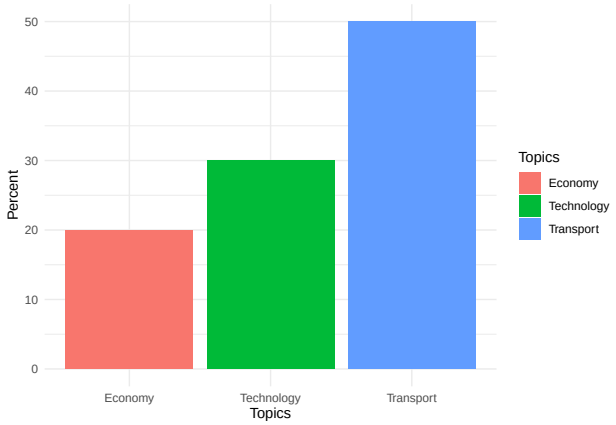


Figure 3: Example of a multinomial distribution of the Topics in LDA for a single document.

Documents have a probability distribution over topics, as described in the example. A document chosen at random may have any percent of n different topics.

Equally, it can be said a topic is a distribution over words. For example, the topic transport can be defined by a list of words that appear in that topic with varying frequencies. For example consider that the topic "transport", for simplicity, comprises three words - car, bus, roads - "Car" appears 70% of the time for that topic, "Bus" 10% and "Roads" 20%. Figure 4 shows a toy example of a word distribution for the topic "Transport" in from Figure 3. The other two topics in Figure 3 would have their own word distributions.

This is analogous to the reality where, if you were to read an article you might discern it's topic by considering the words used in the article and also the frequency of the words used in the article to make the judgment. For example the word "Car" would appear more often than petrol station across all "Transport" documents - leading to the conclusion that the main topic would be indeed "Transport" and not "Energy" which might be the case if the phrase "Petrol Station" were to be more prevalent.

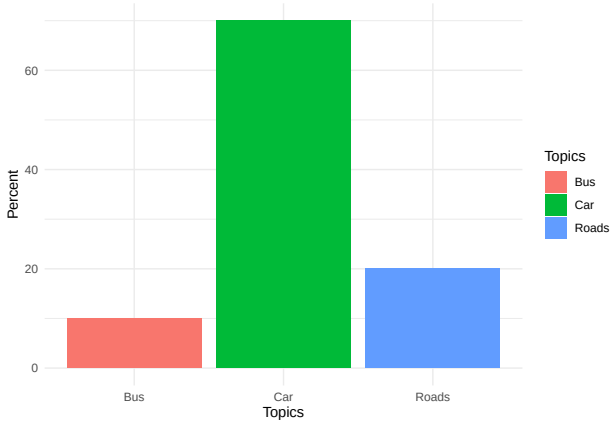


Figure 4: Example of a multinomial distribution of the words in a Topic in LDA.

A topic model is a generative process. The model describes a statistical procedure through which the observed documents are produced. A generative process for documents describes how words in documents might be generated based on variables that are not seen - latent variables. It aims to find the best set of latent variables that explains the documents in the corpus. These latent variables dictate the probability distributions described above in the toy examples.

3.2 The Generative Process of LDA

A document in the corpus can be assumed to be created by first choosing a distribution over topics - just like how an author might decide he is going to write about 60% "transport" and 40% "technology". These two topics are then the distribution of topics per document. After, for each place where a word appears in the document, a word is chosen from a distribution of words from the selected topic; where the selected topic is sampled from the distribution of topics per document. For example, the topic "Transport" is sampled for the first word, and within topic "Transport" there is a distribution of words taking the form of 60% "Car" and 40% "Bus". A word is chosen from this distribution and since car is most likely, for this first word of the document "Car" is chosen. The role of LDA is inferring these topic and word distributions through which the documents most likely have been generated.

In practice when using LDA, each document is a distribution over many topics, but typically each document is dominated by one or two topics - just as in real life where an article typically only has one or two topics. Each of those topics have also a distribution of many words. For example a topic may have thousands of words within its word distribution. This is an important point because, roughly speaking, the Dirichlet distribution is used to constrain these amount of topics that define a document, and the amount of words that define a topic, to a smaller amount. This yields results which are easier to interpret.

This generative process which gives topic and word distributions is useful - by seeing the distribution of words in a topic the topic can be given a name. For example if the words "Fight" and "Glove" were grouped together in a distribution with 80% and 20% probability respectively, it can easily say that this topic is about boxing.

3.2.1 Notation

The distribution over words within the document specified by the modelis given by

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_j = j)P(z_i = j) \tag{2}$$

where $P(z_i = j)$ is the probability that the jth topic was chosen for the ith word, $P(w_i|z_j = j)$ isthe probability of the word wi under topic j and T, the number of topics.

3.3 The Dirichlet Distribution

https://towardsdatascience.com/dirichlet-distribution-a82ab942a879

The Dirichlet distribution $Dir(\alpha)$ is a family of continuous multivariate probability distributions parameterized by a vector α of positive reals. It is a multivariate generalisation of the Beta distribution. The Dirichlet distribution is a conjugate prior to many important probability distributions and specifically a conjugate prior to the multinomial distribution - the distribution LDA uses for sampling topics and words. The Dirichlet distribution is used in LDA to sample a multinomial distribution of topics and words. It is useful because it allows for topics and words to have varying probabilities of being sampled, allowing for documents to be confined to one to two topics which is a more natural way to interpret a document.

Multinomial distributions can be drawn from a Dirichlet distribution which is needed to sample ϕ and θ, where $\phi^{(j)} = P(z = j)$ is a multinomial distribution over words for topic j and $\theta^{(d)} = P(z)$ is a multinomial distribution over topics for document d. These are the latent variables which ultimately LDA aims to approximate. The parameter θ indicates what topics are important for a particular document and ϕ indicates what words are important in each topic.

3.3.1 Toy Example for te Dirichlet Distribution

https://medium.com/@yu.sue.liu example inspired by

Suppose a six-sided dice is to be manufactured with the outcomes of the dice being only the numbers one, two and three. If this is a fair die then the three outcomes with have the same probability of $\frac{1}{3}$. The probabilities of the outcomes can be represented by the vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$. Since the sum of these probabilities must be equal to one and none of the probabilities can be negative, rolling the dice can be described by a multinomial distribution. Since each $\theta_i \in [0, 1]$ this means the set of allowable values for $\boldsymbol{\theta}$ is confined to a triangle. The Dirichlet distribution is used to define the probability density at each point on this triangle.

The Dirichlet distribution defines a probability density for the vector $\boldsymbol{\theta}$ and is given by

$$Dir(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{Beta(\boldsymbol{\alpha})} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \tag{3}$$

where K is the number of variables in the vector $\boldsymbol{\theta}$.

The Dirichlet distribution is parameterised by the vector $\boldsymbol{\alpha}$ which has the same number of elements as $\boldsymbol{\theta}$, that is K. $P(\boldsymbol{\theta}|\boldsymbol{\alpha})$ can be interpreted as giving the probability density associated with the multinomial distribution $\boldsymbol{\theta}$ given the Dirichlet parameter of $\boldsymbol{\alpha}$.

The Dirichlet distribution is a multivariate generalization of the beta distribution. The beta distribution is defined in the interval [0,1] and has two parameters, α and β. This distribution is a conjugate prior for the binomial distribution, hence the multivariate form of the beta - that is the Dirichlet distribution is a prior for the multivariate form of the binomial distribution, the multinomial distribution. In 5 it can be seen how the distribution profile is changed depending on the values of α and β. When these parameters are less than one, it can be seen that the edges of the distribution are regions with high probability density. In the context of topic modelling, this can be useful because the samples from this distribution will be generally from one topic since the extremes of the distribution can be linked to a topic.

The Dirichlet distribution can also be visualised where $K = 3$ - in other words this serves as a toy example where the number of possible topics for the corpus is 3. This means visualising the distribution on a triangle simplex for differing values of α as to investigate how changing α changes the probability distribution.

In 6 it can be seen that when $\alpha_i < 1$, or that is to say $\boldsymbol{\alpha} < K$m high distribution areas are in the corners of the triangle. the corners of the triangle can be thought of as a topic, since when sampling from the Dirichlet a topic is being sampled. As stated above, this is useful because a document is typically made up of one or two topics in the real, not a mixture of all possible the topics as would be the assumption if the Dirichlet distribution with all $\alpha_i = 1$ would be used, as seen in the second triangle in 6. With the SVD

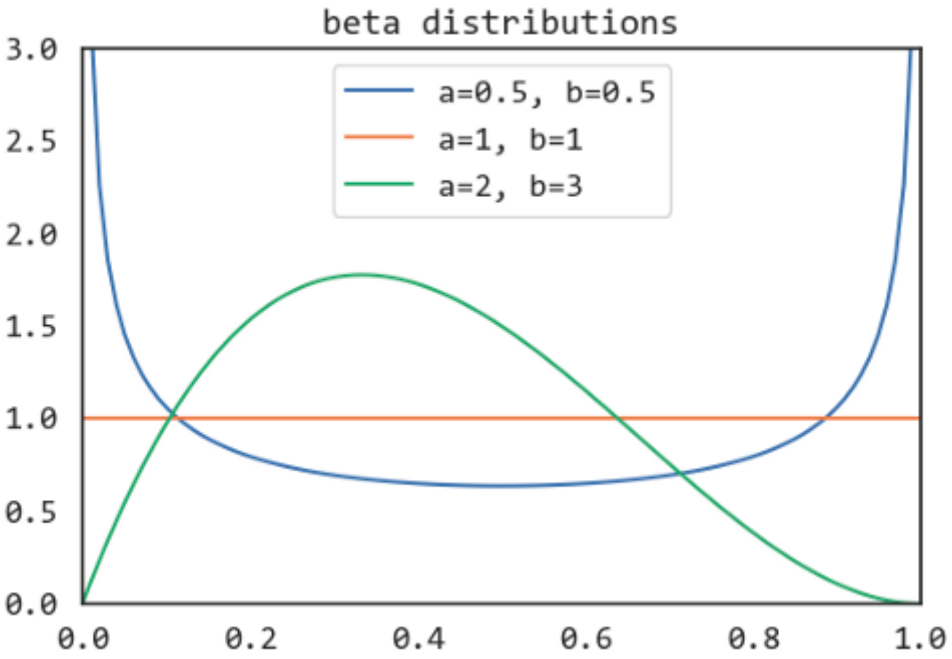


Figure 5: The beta distribution with different parameters

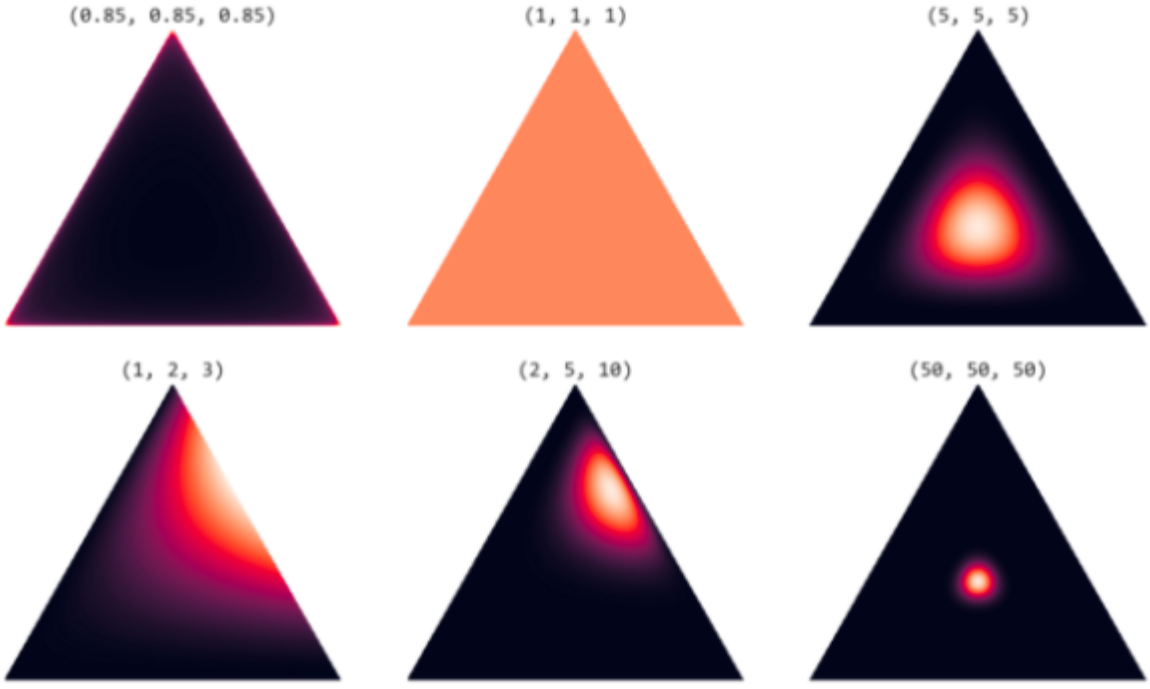


Figure 6: How the probabiliy distribution for the Dirichlet changes with varying values of alpha

Matrix Factorisation approach described before with LSA, each document had a value for each topic, so a document is assumed to be made up of all the topics in the analysis at varying proportions. LDA has the ability to assume a document is made up a small number of topics using the Dirichlet distribution, which is a natural assumption as often pieces of texts only deal with one or two topics, for example, "Technology" and "Science".

3.3.1 Dirichlet in detail

With LDA, there is a Dirichlet prior on $\theta^{(d)} = P(z)$, a multinomial distribution over topics for document d. This is a conjugate prior for the multinomial. The probability density of a T, the number of topics, dimensional Dirichlet distribution over the multinomial distribution $p = (p_1, ..., p_T)$ and is given by:

$$Dir(\alpha_1, ..., \alpha_T) = \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_{j=1}^{T} p_j^{\alpha_j - 1} \tag{4}$$

The parameters of this distribution are specified by $\alpha_1, ..\alpha_T$. Each hyperparameter α_j can be interpreted as a prior observation count for the number of times topic j is sampled in a document before having observed any actual words from that document. The Dirichlet prior on the topic distribution θ results in a smoothed topic distribution depending on the α parameter. This parameter controls the number of topics for each documents or in other words it controls the probability distribution of topics for each document. Sampling from the Dirichlet distribution means getting a multinomial distribution for the topics for that document. The α is a hyperparameter, that is a parameter of parameters, that dictates what the multinomial distribution of topics looks like for each document. The benefit of the Dirichlet distribution is that for $\alpha < 1$ the multinomial distribution is often represented by a few topics. If $\alpha > 1$ the distribution is quite even - that is the distribution is represented by many topics. It is more natural to assume a document only contains a few topics, just like a document in real life. For that reason an $\alpha < 1$ is used in topic modeling.

A Dirichlet distribution prior is also placed on ϕ, the multinomial probability distribution of words of each of the topics, using the parameter β. It can be interpreted as the prior observation count on the number of times words are sampled from a topic before any word from the corpus is observed. Just like the α parameter, it controls the word distribution for each topic, either making the probability distribution uniform or having only a few words have high probabilities for each topic.

In summary:

The Dirichlet distribution has two parameters:

- α - controls how different the probabilities will be for words in the topics.
- β - controls how different the probabilities will be for topics in the document.

Plate notation

Figure 7 shows a way to graphically represent the LDA model. Shaded and unshaded areas indicate observed and latent variables respectively. ϕ, the distribution of words in the topics, θ, the distribution of topics in each document and z, the assignment of word tokens to topics are what three latent variables aim to infer. α and β are the hyperparameters that are treated as constants in the model. Arrows represent conditional dependencies between the variables and the plates, the boxes in the figure, refer to repetitions of sampling steps with the variable in the right corner referring to the number of samples. For example, the inner plate that has the bubbles z and w illustrated the repeated sampling of topics and words until N_d, the number of words in each document d, words have been generated for document d. The plate surrounding $\theta^{(d)}$ shows the repeated sampling of a distribution of topics for each document d for a total of D documents. The plate surrounding $\phi^{(z)}$ shows the repeated sampling of word distributions for each topic z until T documents have been generated.

Here is a labeled version of Figure 7

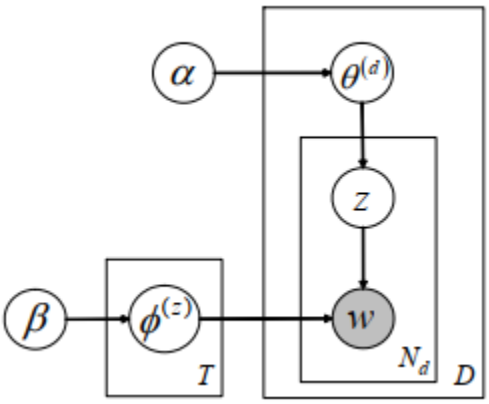


Figure 7: Box diagram

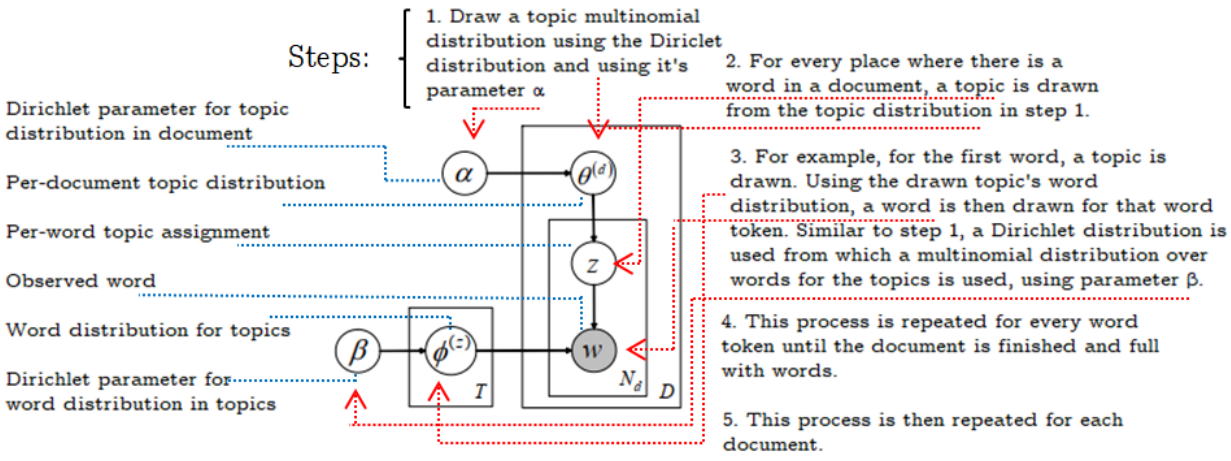


Figure 8: Box diagram labeled

The goal of LDA is to estimate θ and ϕ. These are latent variables that aren't observed that the model attempts to fit. These latent variables are determined by using the words observed in the corpus, so the document document making process is reverse engineered to get the parameters that make those observed words. LDA aims to find the parameters that maximise the likelihood of the words observered in the corpus.

Unlike statistical models like OLS, there is no analytic solution with LDA. The best solution can be approximated in an iterative way, using steps and getting closer to the best solution. One way to do this is with Gibbs Sampling.

3.4 Gibbs sampling

Here is the intuition behind Gibbs sampling. Assume it is known that the topics of all the words except for the word token of interest. For this word token of interest, a topic must be chosen from the topic distribution for the document. The model prefers to choose a topic that is already present in the document. At the same time, the model also wants to pick a topic that the word already occurs in. So there are 2 probability distributions: one that favors the topics already present in the document and one that favors the topics that this word is already present in.

Taking these two probability distributions together, a joint probability distribution is gained and will give you the chance of picking a topic for this one missing word.

Then, using the iterative Gibbs Sampling in this way:

1. Start with random sampling - assigning every word in every document to one topic.
2. Then iterate over all the words and all the documents. For each word in each document, the proportion of topics is computed in that document (disregarding the word itself).
3. Compute the proportion of topics for that word (disregarding the word itself). I.e what are the topics that the word occurs in and at what frequency?
4. Multiply these two distributions to get a joint probability distribution.
5. Choose a new topic from that probability distribution and then update the proportions.
6. Repeat this step, going to the next word, and eventually to the next document, until every word is updated in every document.
7. Initially there were completely random distributions, but after these steps there are slightly less random distribution because the this method, for the words, tends to choose topics that are consistent with that document.
8. Each time these steps are run a stable solution is moved towards, and these steps are run until it converges to a stable solution.

3.4.1 Gibbs Sampling in detail

Gibbs sampling is used. This is a form of the Markov Chain Monte Carlo sampling method which samples values from high-dimensional distributions. Gibbs sampling simulates a complex distribution by sampling over marginal distributions given a specific value of the other distribution. This is done sequentially until the sampled values approximate the desired distribution.

The Gibbs procedure considers each word token in a document and estimates a probability of assigning the current word token to each topic, conditioned on the topic assignments on the topic assignments to all the other word tokens. From this conditional distribution, a topic is sampled and then is the new topic assignment for this word token. This conditional distribution can be written as:

$$P(z_i = j \mid z_{-i}, w_i, d_i,) \tag{5}$$

$z_i = j$ represents the topic assignment of token i to topic j, z_{-1} refers to the topic assignments of all the other word tokens and the dot refers to all other known or observed information as such as all the other word and document indices w_{-i} and d_{-i} and hyperparamters α and β.

This probability is proportional to

$$P(z_i = j|z_{-i}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_i j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i j} + T\alpha} \tag{6}$$

where C^{WT} and C^{DT} are matrices of counts, C^{WT}_{wj} contains the number of times word w is assigned to j, not including the current iteration of i, C^{DT}_{dj} contains the number of times topic j is assigned to some word token in document d, not including the current iteration i.

Note that equation (4) isn't the actual probability of assigning a word token to topic j but it is proportional to the right hand side. The actual probability is produced by dividing the equation by the sum of all topics T. The left part of the right hand side of the equation can be seen as the probability of word w under topic j. The right part is the probability that topic j has under the current topic distribution for document d.

Because the top of the left part C^{WT}_{wj} contains the number of times word w is assigned to j, it means that when many tokens of a word have been assigned to topic j across the documents, it will increase the probability of assigning any particular token of that word to topic j.

At the same time, looking at the right part of the right hand side, if topic j has been used multiple times in one document, this means a higher C^{DT}_{dj}, the number of times topic j is assigned to some word token in document d, it will increase the probability that any word from that document will be assigned to topic j.

As a result, words are assigned to topics depending on how likely the word is for the topic and also how dominant that topic is in a document.

1. Assign each word token to a random topic.
2. Decrease the count matrices C^{WT} and C^{DT} by one for the entries that correspond to the current topic assignment.
3. A new topic is sampled from the distribution in Equation 4 and the count matrices C^{WT} and C^{DT} are incremented with the new topic assignment.
4. Repeat these steps for all N word tokens in the corpus, achieving one Gibbs sample.
5. Repeat steps 1-4 until a stable solution converges - that is the posterior distribution over topic assignments are approximated as best as possible.

Obtaining the topic and word distributions.

Recall that:

- $\phi^{(j)} = P(z = j)$, a multinomial distribution over words for topic j
- $\theta^{(d)} = P(z)$, a multinomial distribution over topics for document d

These can be obtained as follows:

$$\phi'^{(j)}_i = \frac{C^{WT}_{ij} + \beta}{\sum_{k=1}^{W} C^{WT}_{kj} + W\beta} \tag{7}$$

and

$$\theta'^{(j)}_j = \frac{C^{DT}_{dj} + \alpha}{\sum_{k=1}^{T} C^{DT}_{dk} + T\alpha} \tag{8}$$

3.5 Benefits of LDA

LDA adds a Dirichlet prior on top of the data generating process, meaning NMF qualitatively leads to worse mixtures. It fixes values for the probability vectors of the multinomials, whereas LDA allows the topics and words themselves to vary.

Thus, in cases where it is believed that the topic probabilities should remain fixed per document (oftentimes unlikely) — or in small data settings in which the additional variability coming from the hyperpriors is too much — NMF performs better. Otherwise, LDA is more suitable to use.

4. Aims

4.1 Motivation for and summary of the project

In early 2021, the impact of social media on the stock market gained notoriety when the price of GameStop (GME) stock went from hovering around $20 to reaching the lofty heights of $347 dollars in just one month. This increase was attributed to the collective frenzy that took place on the social media platform of Reddit, specifically the sub-forum known as "wallstreetbets" that encouraged users to buy more stock which in turn elevated the stock to those high prices.

This was a clear demonstration of how public word-of-mouth can impact the stock market, but this kind of thing has happened for time immemorial. Before social media, word-of-mouth opinions about stocks would have also made an impact on the price of stocks. The difference now is these opinions are much more visible, and important for this project, they are recorded.

This project aims to use Topic Modeling on Tweets that refer to the company Gamestop and the cryptocurrency Bitcoin. Using Topic Modeling, tweets are classified into groups depending on the topic discussed in each tweet. It further looks to investigate how these topics of public opinion influences the price of the assets Gamestop and Bitcoin.

4.2 An overview of the approach to be taken

There are two main steps when it comes to approaching this analysis.

1. Assigning topics to tweets that represent opinions about stocks and cryptocurrency.
2. Finding the relationship between these opinions and change in asset price.

1. Classifying posts/articles

Reading social media posts gives the reader an understanding of the opinion and feeling of the public for a particular topic. For example, a social media poster may simply say *"I believe Bitcoin is going to be worth £1.2 million one day* and although the poster's believed price may not be remembered by the reader, the sentiment of the post - that bitcoin is going to increase in value - will be.

In this example for simplicity and demonstrative purposes, consider trying to assign tweets to just two topics - "positive" or "negative"; the above example tweet can be classified as a *"positive"* sentiment for Bitcoin. A post using skeptical language in regards to bitcoin can be classified as *"negative"*.

A *Probabilistic Topic Modeling* which was applied to machine learning and outlined by David Blei et al can be used to analysed posts/articles for the type of language they use regarding a certain asset at a time point, and the frequency of positive or negative posts can found for a particular asset. This then gives us a "sentiment score" - telling what is the public opinion of the asset, is it mainly positive, negative, or equal? This type of analysis is known as *Natural Language Processing (NLP)*.

Classifying a post/article into positive and negative is a useful way to gauge public opinion, however, positive and negative posts can be classified into further topics. For example, how positive is the post, perhaps the language of the post is hugely positive, so it can be classified as *"very positive"*. Perhaps the language used is positive, but tinged with some negativity of skepticism - and this could be classified as *"weakly positive"*. Further still, within the trading community there are trading behaviours known as *bearish* (many people following the trend of selling) and *bullish* (many people following the trend of buying) and the posts can be further classified into these groups.

To summarise the above, the use of Topic modeling algorithms analyses the words of the original texts to uncover the underlying topics that run through the tweets and how the importance of these topics change

through time depending on when the tweet was posted. This project uses uses Latent Dirichlet Allocation (LDA) to allocate tweets to topics.

4.3 The Data

Tweets were pulled using the Twitter API. Three hundred tweets were pulled for each day that contain the terms "BTC" or "Bitcoin" for Bitcoin data, and the terms "GME" or "Gamestop" for the Gamestop data. This data was pulled for tweets posted from Janurary 1st 2016 until August 31st 2021. The data is extracted as a CSV (comma-separated values) file.

This data will then be run through an LDATopic Modeling pipeline to find the underlying topics in these tweets.

5. Pipeline Architecture

All code was written and executed using Python 3.7.

5.1 Cleaning

Before an LDA model can be built the tweets must be "cleaned". This cleaning process is done to remove noise from the tweets so that the model can uncover the meaning of the tweets more effectively. Before this cleaning process is undertaken all characters in the corpus are converted to lowercase and all subsequent filters act independently on the character's case. Uppercases are used to describe the filters in this document to fit within the grammatical rules of producing this report only.

There are two main cleaning steps that were undertaken on the data.

1. Corpus-wide noise: the removal of tweets that are not relevant to the analysis. Examining the tweets uncovered a portion of tweets that don't express opinions of people and are most likely tweets by bots. Here is an example tweet

   "Current price: 433.54$ $BTCUSD $btc #bitcoin 2016-01-02 18:40:05 EST"

This type of tweet appears multiple time in the dataset and more than likely is produced by a bot that serves to update users of twitter the price of bitcoin. There are many tweets such as these that can judged as being produced by bots and removing them from the data is important because they don't represent any sentiment of actual humans. These tweets are considered noise and are removed.

These tweets were found by running the LDA algorithm on the corpus and exploring the t-SNE plot which shows individual tweets in each topic. This plot will be discussed later.

If the tweet contained one of the phrases from the following list they were removed from the corpus and deemed to be an irrelevant tweet. This does not capture all irrelevant tweets and may delete tweets that were posted by a human (each filter works independent of whether the characters are upper or lower case):

- Current price
- Price
- Price update
- Prices update
- Price action
- Price increase
- Price decrease
- Density
- Last hour
- Latest block info
- Closed sell
- Alert
- Hourly update

- %
- Removal of tweets that contain ten or more digits. These tweets were observed to be produced by bots reporting statistical data about bitcoin.

The removal of these tweets means that the sentiment of the users of Twitter can be more effectively investigated.

2. Removal of within document noise. For example, a word which appears in the data may be presented like this "#hate". This phrase carries the same meaning as the word hate phrased as this "hate". In this example, the hashtag is removed from "#hate" and the word is saved in the tweet as "hate". As explained previously, Topic Model algorithms depend on the frequency of words in the corpus. It would not be accurate to consider "#hate" and "hate" as two separate frequencies.

For the statistical analysis, "#hate" and "hate" will be grouped together because they are exactly the same in terms of their sentiment, and this is important because topic modelling's purpose is to group tweets based on similarities within those tweets and the way to pick up that similarity is by seeing if tweets have the same words inside them. The algorithm has no way of knowing if "#hate" and "hate" are the same word. For that characters such as hashtags are removed from the tweets, amongst other characters that will be explained further.

The following cleaning of characters were undertaken for each tweet. using regular expressions:

- Conversion of html escapes to characters, for example conversion of "&" to "&".
- Removal of html tags. For example "" is .
- Removal of markdown URLs, for example: Some text
- Removal of text or code in square brackets.
- Removal of sequences of special character, for example "&#" is removed.
- Removal of sequences of hyphens such as — or ==
- Removal of sequences of white spaces
- Removal of #
- Removal of $ (this charatcer is used as a cashtag, similar to a hashtag)

Using the python package "Textacy" normalisation was executed on the corpus. Normalisation is a process that transforms text into its canonical form. For example, the word "bóat" with an accent would be converted to "boat" without an accent.

- Normalize words in the document that have been split across lines by a hyphen for visual consistency (aka hyphenated) by joining the pieces back together, sans hyphen and whitespace.
- Normalize all "fancy" single- and double-quotation marks in text to just the basic ASCII equivalents.
- Normalize unicode characters in the document into canonical forms.
- Remove accents from any accented unicode characters in text, either by replacing them with ASCII equivalents or removing them entirely.

The following words are removed from each document as they are synomyms for the word Bitcoin and don't carry any sentiment. The decision to remove these was made by observing the result of the topic model and observing these words having high importance in topics:

- BTC
- Bitcoin
- Crypto
- Cryptocurrency
- Coin

5.1.1 5. Remove "Stop words"

These are commonly used words in English that don't carry little to no information. Words such as "The" or "a" are filtered out. This removes noise from the data in order for the algorithm to give more focus to the more meaningful words.

5.1.2 Lemmatisation

In addition to removal of special characters to ensure words with the same meaning are the same, the syntax of a word is cleaned. A short example of this is, the phrase "buy" and "bought" carry the same meaning - they just refer to different moments of time, the present and past respectively. For that reasons these words should appear the same for the algorithm as they carry exactly the same sentiment and meaning; they will be both classed as the present "buy". This helps the algorithm give the same meaning to tweets that use words that only differ by their grammar (such as past, present, future, gerund, past participle). It seeks to make each word be independent of its grammatical version.

5.1.3 Tokenisation

This process is the separation of the tweet into single words, and these words are known as "tokens", so the LDA algorithm can determine where each word starts and ends. This is necessary for the algorithm to analyse the data.

5.1.4 n-grams: Including phrases

This joins two or three (bigram or trigram, respectively) tokens together to form a two or three token phrase. If a two or three word phrase appears of often in the tweets, specifically if they appear more than 30 times, they are included in the dictionary which is used in the LDA algorithm. This is important because some words when joined together contain a very different meaning. For example, the phrase "money-hungry" gives a different meaning to the document than if the words "money" and "hungry" were to be used separately.

5.1.5 Filtering tokens that exhibit extreme frequencies

If a word appears only 5 times in the whole corpus, it is removed from the dictionary. If a word appears in eighty percent or more of the tweets, it is also removed from the dictionary. Since these words are removed, they are then not used in the LDA algorithm.

Words that are very frequent or not very frequent are not relevant to the analysis because if they appear very frequently, then they're not a good way to distinguish between topics. If they don't appear frequently documents won't have these words in common and simply adds to the noise of the data.

5.2 Further data pre-processing steps

1. Create a dictionary - This is the list of all the words in the corpus (all the documents).
2. Create Bag of Words for each document - This is gives a unique number ID to each word in each tweet, accompanied with the frequency of that word in the document. This is necessary since the algorithm will process the documents using ID number of each words and their frequencies.

5.3 Running the LDA Algorithm

Using two packaes known as Gensim and LDA Mallet, the model is run through the LDA algorithm using the pre-processed data and using a β parameter of 0.01 and an α parameter of $\frac{5}{T}$ where T denotes the number of topics specified in the algorithm.

5.4 Evaluatiing the Best Number of Topics

The LDA model requires that the statistician inputs the number of topics that the corpus should be split into. Since the actual number of topics is hard to discern from such a large corpus of tweets, way to mathematically ascertain the best number of topics is necessary. Choosing the right number of topics for the model to output is important because if too many are chosen those superfluous topics won't make sense. Equally if too few

are chosen the model would miss topics which might be important to the statistician. Topic models make no guarantee on the interpretability of their output nor their topics.

The overall aim of evaluating a topic model is seeing if the outputted topics tell a good story and if the topics are well-defined. One way of doing that is called a "Word Intrusion" test. This is a test based on a human observers. For each topic, the list of the most probable words in that topic are presented to an observer, with one important word in that topic substituted with a word that has a high-probability from another topic.

Given is a toy example that describes the "Word Instruson" test

Original Topic top 5 most important words : car, boot, window, drive, road Original Topic with Word Intrusion: car, boot, banana (word intruder), window, drive, road

If the human observer can identify the word intruder, then the original list of most important words for that topic makes sense and therefore the topic is well defined. If the observer can't identify the intruding word then the original topic's list of most important words doesn't make sense because it means the word intruder looks as good as the words in the topic before, which suggests that the topic's important words are defined at random. Tt suggests that the number of topics the user chose as a parameter should be changed.

This Word-Instrusion test is a measure of topic "Coherence". Coherence in this context is a measure of how much the documents support each other - or in other words how well the topics resemble an actual topic that human labels it as a distinct topic. These human-based measures serve as a gold standard for coherence evaluation. However, they are expensive to produce and therefore a statistical based measure is called for.

There are several popular coherence measures used in Topic Modeling with UMass being a popular metric. This measure correlates well the the human based word intrusion measure. UMass calculates the correlation of words in a given document based on conditional probability. It takes the set of N most important words of a topic and sums a **confirmation measure** over all the word pairs in the set. This confirmation measure takes a pair of words or word subsets as well as the corresponding word probabilities to compute how strong one set of words supports the other. This score measures how much, within the words used to describe a topic, a common word is on average a good predictor for a less common word. The confirmation measure is given by:

$$C_{UMass} = \frac{2}{N \cdot (N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i=1} log \frac{P(w_i, w_j) + \epsilon}{P(w_j)} \tag{9}$$

Ths sum of this gives the UMass Coherence score. Models are run for each unique value of number of topics $k \in [5, 45]$, where k is the number of topics. It is computationally expensive to run the models therefore the number of topics has been constrained to this set. The model with the lowest Coherence UMass scored is considered as the best model with k being the best number of topics.

https://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf [temporary]

A paper (above) compared many different coherence measures by looking at their correlations with human based measures of topic models. The best performing coherence measure was found to be newly proposed measure. This measure, C_V combines the indirect cosine measure with the NPMI and the boolean sliding window. See https://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf for more detail.

This score ranges from zero to one with one being being perfect coherence. Like the UMass score, multiple models will be run for each value for the number of topics and the value closest to one will be considered as the best number of topics to use. Although it is also important that the topic makes sense for a human and therefore these topics were analysed such that words in the topics were judged by the human to belong together.

6 Visualisation

6.1 LDAvis

Visualizing the result of a topic model is difficult since LDA since the fitted model has many dimensions - the LDA model is applied to thousands of documents, modeled as many topics, and these topics are modeled as distributions over many words. Using a pakacge called pyLDAvis, the result of the topic model can be visualised in a user-friendly way.

The left pane of the visualisation shows the whole view of the topic model, showing how prevalent each topic is and how the topics relate to each other. Each topic is a circle in a two-dimensional plane. Their positions are determined by computing the distance between topics and also using multidimensional scaling to project the inter-topic distances onto two dimensions (done in (Chuang et al., 2012a)). The prevalence of a topic is represented by the area of its circle.

The right hand panel shows a bar chart, showing the most "useful" words for a topic highlighted on the left panel. This intends to show the meaning of the topic. The bars show two overlaid bars per word - the topic-specific frequency of a word (red) and the corpus-wide frequency of the word (blue). Selecting a word on the right changes the size of circles of the topics on the left - with the sizes of the circles representing the conditional distribution over topics of the chosen word.

6.1.1 "Useful" words

In order to observe a topic meaning, a ranked list of the most probable terms in that topic. The issue with investigating topics like this is that common words in the corpus occur in the ranked list of multiple topics which in turn makes it hard to differentiate meanings of these topics. In order to resolve this issue Bischof and Airoldi (2012) suggested ranking terms for a given topic by using the frequency of the term in that topic and the exclusivity of the term to the topic. The exclusivity takes into account how many times the term appears in that topic to the exclusion of others. This metric is defined as "usefulness" and pyLDAvis uses a similar measure called "relevance" to rank terms in order of "usefulness". Relevance is defined as:

$$r(w, k|\lambda) = \lambda log(\phi_k w) + (1 - \lambda) log(\frac{\phi_k w}{p_w}) \tag{10}$$

where $\phi_k w$ denotes the probability of term $w \in \{1, ..., V\}$ for topic $k \in \{1, ..., K\}$ where V denotes the number of terms in the vocabulary. Also, p_w denotes the marginal probability of term w in the corpus. λ determines the weight given to the proability of term w under topic k relative to to its lift. Lift is defined as the ratio of a term's probability within a topic to its marginal probability across the corpus, which can be seen as the subject of the log function. Setting $\lambda = 1$ results in ranking the terms in a topic by their topic-specific probability. Setting $\lambda = 0$ ranks the terms soley by their lift.

A study was carried out with humans to see what the optimal value of λ was. (temporary, from here https://nlp.stanford.edu/events/illvi2014/papers/sievert-illvi2014.pdf). Using documents that have been assigned topics (ground truth), subjects were asked to assign a list of words to a topic - with the list of words coming from topics made from a topic model. The list of words presented to the subjects were made with varying values of λ and the relationship between accuracy rate assigment of the list of words to a topic and values of λ was explored. The study found that the optimal value fo λ was 0.6 (no confidence interval given) yielding 70% probability of correct topic identification.

6.2 t-SNE

Another useful visusaltion of topic models is known as t-SNE. It stands for "t distributed Stoachastic Neighbourhood Embedding" and was developed in 2008. This is an unsupervised non-parametric method of dimensionality reduction, similir to the left hand pane of the LDAvis. It's useful in separating data that cannot be separated by any straight line. "t-SNE" preserves the local structure in the data suc hthat points

that are close together in high dimensions will also be close together when the dimensions are reduced by the algorithm to just two dimensions.

It's useful for visualisation since, with Principal Component Analysis (PCA), a popular dimension reducing algorithm, the global structure of the data is preserved. PCA concerns itself with the direction in which the variance is maximises and it doesn't take into account the distance between individual points. t-SNE on the other hand concerns itself with the local structure and preserves the both the local and global structure of the data. This is importance when displaying topic models since it makes sense that similar documents be close together. It's also good art handing non-linear data. PCA is a linear algorithm; creating components which are the linear combination of existing feature. This means it's not able to interpret polynomial relationships between features. t-SNE is capable of working with non-linear data however in t-SNE is computationally complex and expensive.

7. Results

7.1 Best Coherence Score

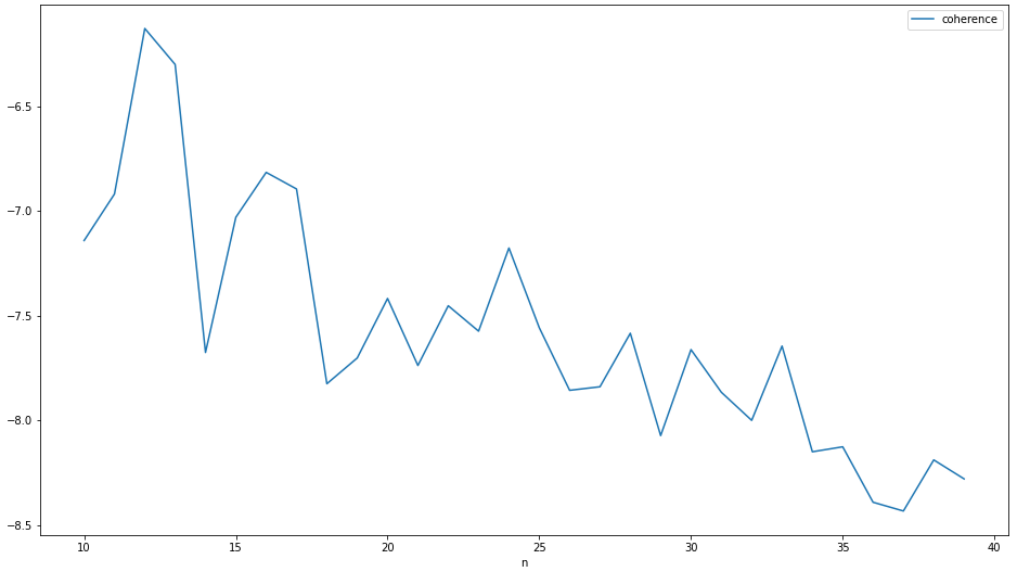


Figure 9: Plot of UMass Coherence score over the number of topics outputted

C_V shows best score is 18

7.2 Topics with the most important words

7.3 Example documents for each topic

7.4 Word Clouds

images/03 - Topic_0

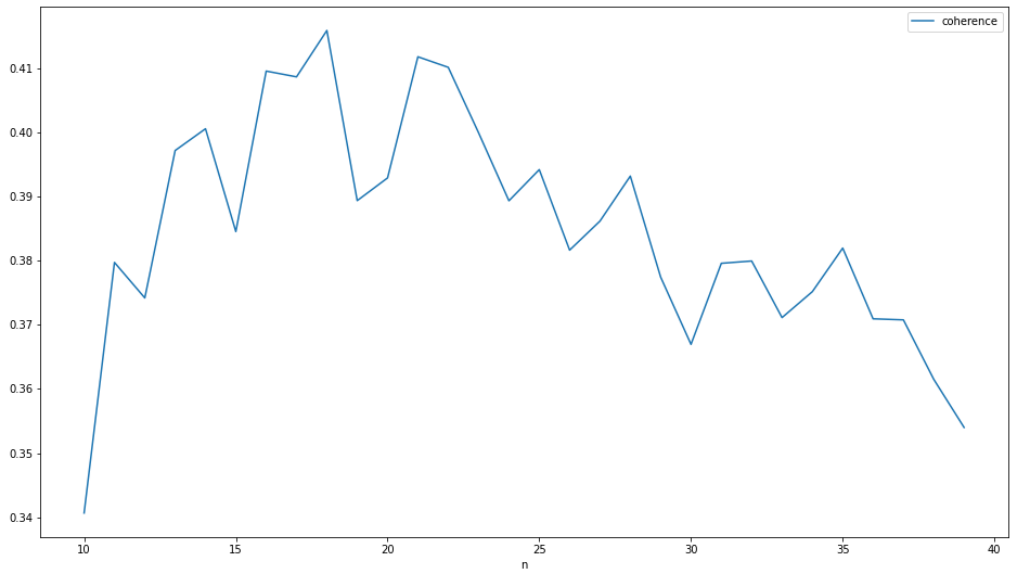
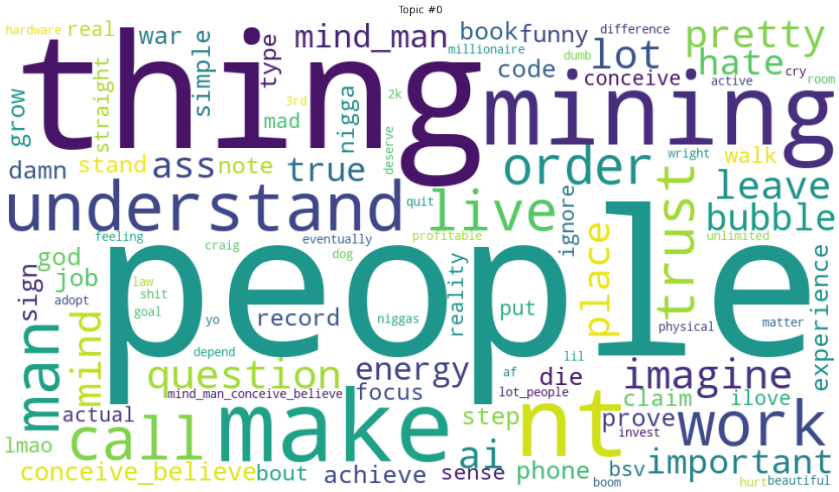


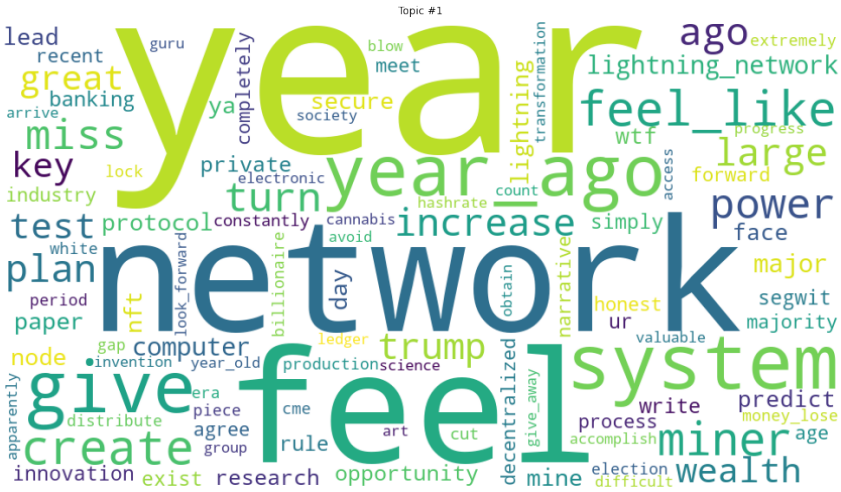
Figure 10: Plot of UMass Coherence score over the number of topics outputted

Table 1: Topics and their top 30 most important words, ordered by importance

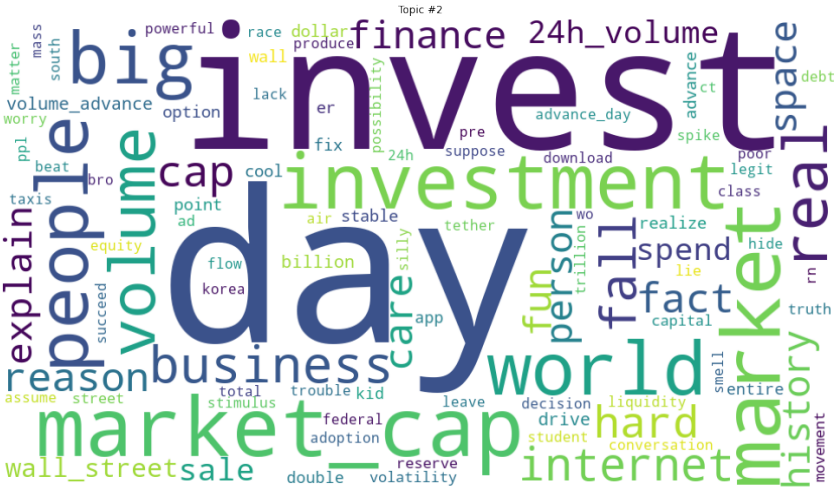
Topic	Term
Topic 0	people, thing, mining, make, nt, understand, man, work, call, live, trust, order, question, mind, imagine, lot, lea
Topic 1	year, network, feel, system, give, year_ago, feel_like, create, miner, power, miss, turn, ago, large, increase, plan
Topic 10	buy, gon_na, sell, shit, change, buy_dip, life, na, gon, lol, friend, dip, fuck, change_life, invest, fucking, success
Topic 11	win_play, long, play, game, win, run, stop,long_term, bull, bull_run, expect, set, easy, continue, moment, char
Topic 12	good, look_like, day, low, month, break, high, stock, guy,stock_market, chart, move, reach, nice, ath, month_l
Topic 13	hour, cash, fee, send, transaction, wallet, rate_sit, rate, block, sit, sell, hour_fee, vbyte,broadcast_transaction,
Topic 14	time, hit, happen, end,bear_market, watch, drop, bear, big, ready, hope, remember, rise, high, toggle_rig, suffe
Topic 15	start, pay, accept, payment, blackjack,blackjack_blackjack, fast, free, million, developer, find, paypal, card, tha
Topic 16	pump, dump, profit, follow, satoshi, lot, community, learn, early,satoshi_nakamoto, bring, build, share, find, pa
Topic 17	currency, market, trade, doge, bullish, scam, idea, doge_doge, current_value, market_1h, signal, usd, exchange
Topic 2	day, invest, market_cap, market, investment, world, big, people, volume, real, business, fall, internet, hard, cap
Topic 3	blockchain, ethereum, currency, ico, project, digital, token, asset, eth, work, technology, team, smart_contract,
Topic 4	money, today, marine_mtc, lose, marine, time, alt, bad, currency, mtc, buy, lose_money, open, dollar, convert,
Topic 5	news, love, worth, talk, twitter, close, rt, account, number,currently_worth, usd, daily, store_value, usd_goldja
Topic 6	alt, eth, xrp, currency, ethereum, ltc, lite, ripple,bull_market, eth_ltc, bch, crash, economy, moon, ada, hodl, g
Topic 7	price, current_price, wait, prediction, current, join, trend, trend_prediction, status, min, channel, sec, pm, tele
Topic 8	exchange, future, usd, bank, gold, exchange_usd, currency, average_price, world, fiat, average, government, ba
Topic 9	short, week, trading, hold, tweet, short_term, investor, trader, level,play_expert, term, add, base, bag, fund, ca



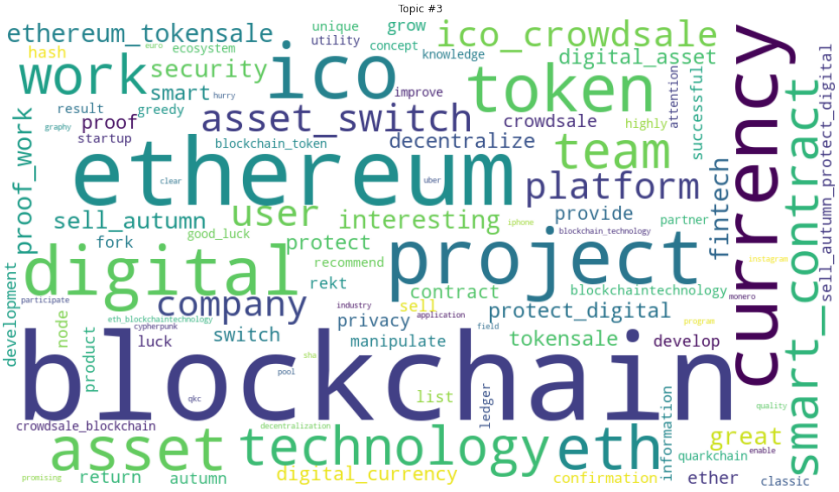
images/03 - Topic_1



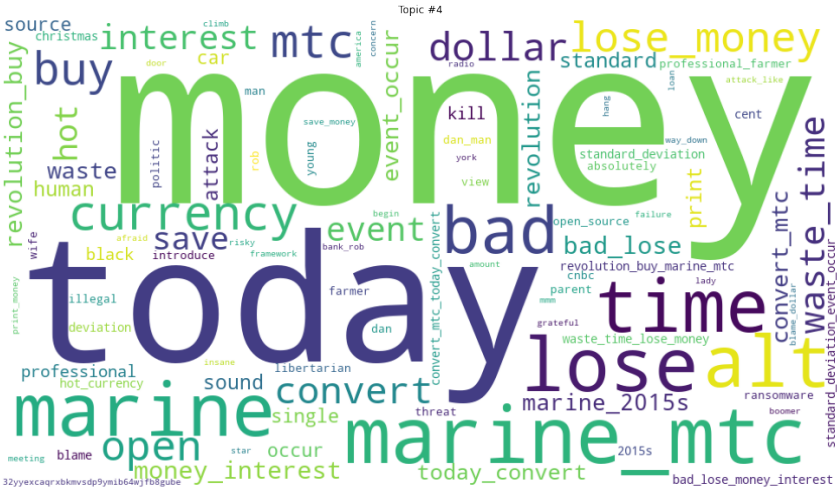
images/03 - Topic_2



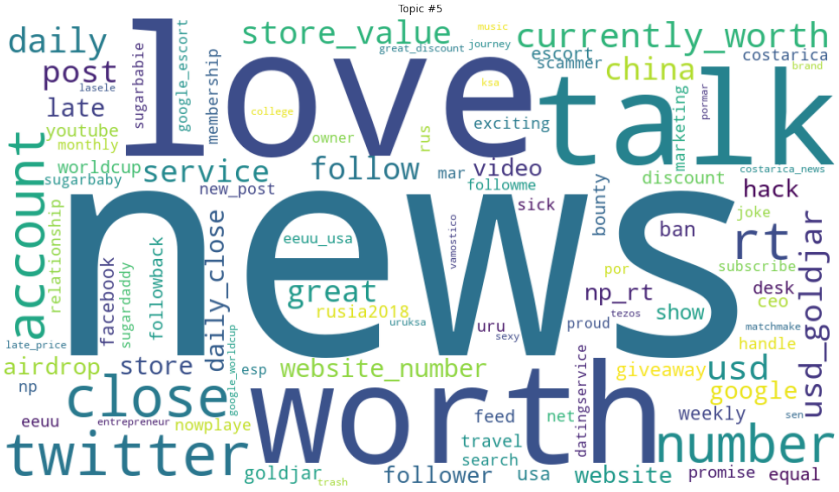
images/03 - Topic_3



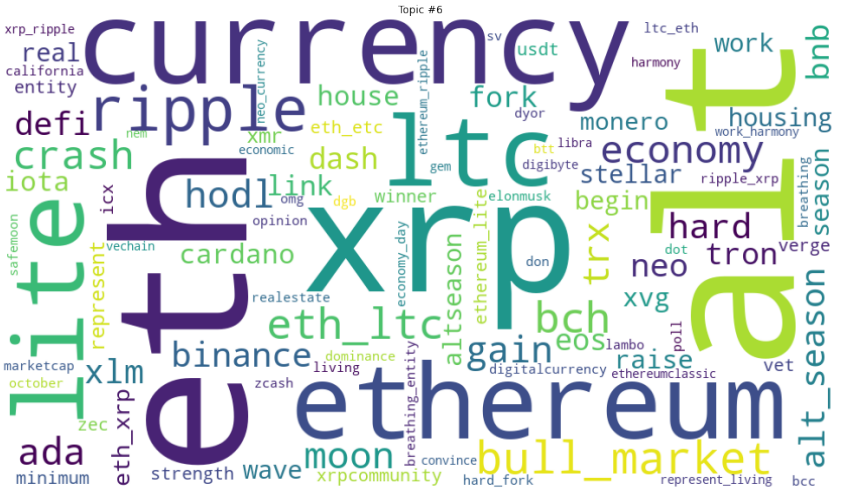
images/03 - Topic_4



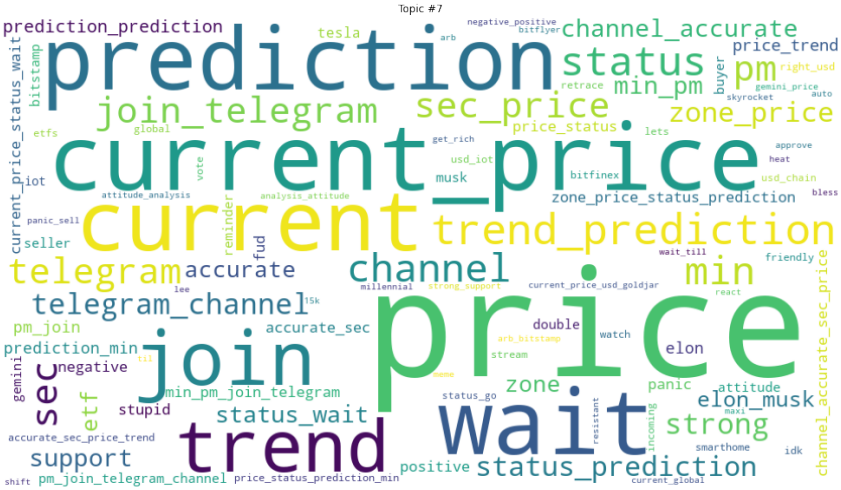
images/03 - Topic_5



images/03 - Topic_6



images/03 - Topic_7

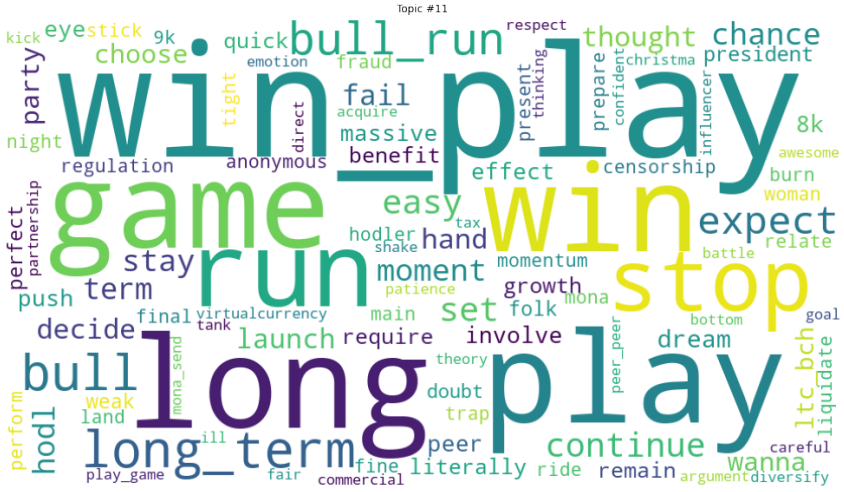


images/03 - Topic_8

images/03 - Topic_9

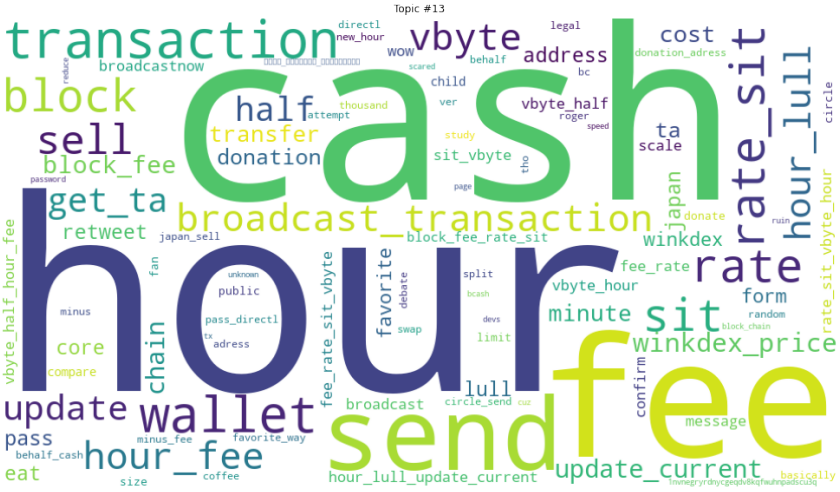
images/03 - Topic_10

images/03 - Topic_11



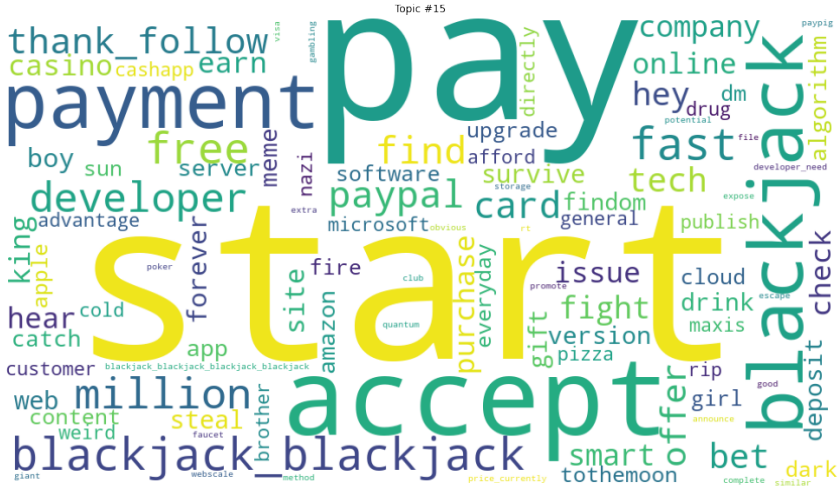
images/03 - Topic_12

images/03 - Topic_13

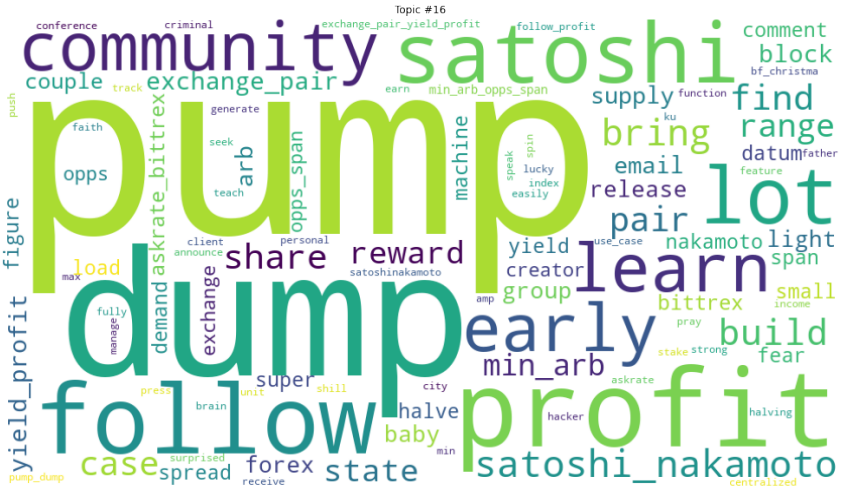


images/03 - Topic_14

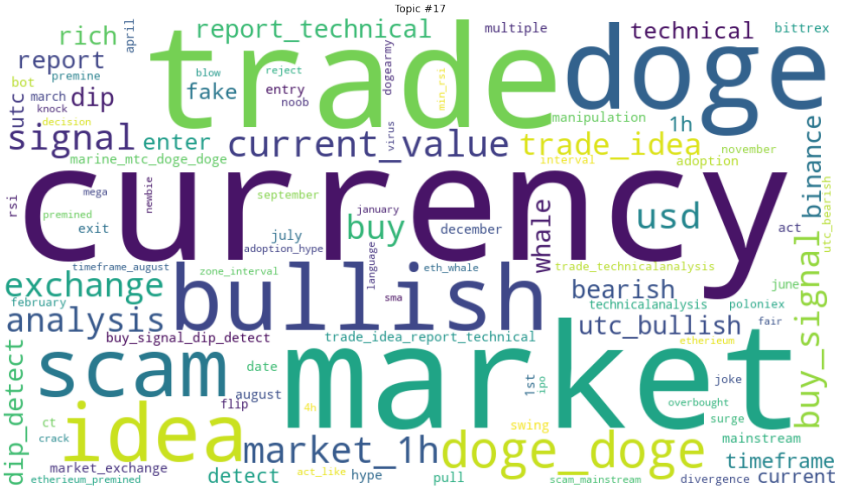
images/03 - Topic_15



images/03 - Topic_16



images/03 - Topic_17



RUBBISH - NOTES

https://nlp.stanford.edu/events/illvi2014/papers/sievert-illvi2014.pdf conducted a user study to determine whether there was an optimal value of lamba from the definition of relevance. r

pyLDAvis

DONT USE A high UMass measure would indicate that the topics are well defined.

Another coherence measure was proposed https://svn.aksw.org/papers/2015/WSDM_Topic_Evaluation/public.pdf which is a unifying framework which combines different coherence measures into one score. It works as follows.

1. **S - Segmentation** : Divide a word set (such as the top most probable words in a topic) into smaller pieces, such as a set of pairs. This is a set of different kinds of segmentations
2. **M - Confirmation Measure** : The set of conffirmation measure that scores the agreement of a given pair, for example UMass
3. **P - Word probabilities** : These can be computed i ndifferent ways so P serves as a set of ways the word probabilities are calculated.
4. **Epsilon - Aggregate**: This is a set of aggregration functions which determine how to aggregate scaalr values.

DONT USE END

There are several methods to use statistical measures to determine the ideal number of topics. One such method is known as the Coherence Score, where a high score represesents.

To find the best Coherence Scrore, a model was built for each n, number of topics. For example, to investigate the best number of topics, a model is first built using the parameter n = 3, 3 topics, and the coherence score is obtained. This process is repeated for n = 4, up until n = 30 and the highest coherence score amongst these models indicates the ideal number of topics.

Cv has issues, as stated herehttps://palmetto.demos.dice-research.org/ 2 issues https://github.com/dice-group/Palmetto/issues/13

i think umass doesnt use wikipedia, but hte rest doesnt. so that makes me think dont use the wikipedia one because these topics are so granular http://qpleple.com/topic-coherence-to-evaluate-topic-models/

https://stats.stackexchange.com/questions/375062/how-does-topic-coherence-score-in-lda-intuitively-makes-sense more explanation

RUBBISH END

References