2017

Triple Non-negative Matrix Factorization Technique for Sentiment Analysis and Topic Modeling

Alexander A. Waggoner
Claremont McKenna College

Claremont Mckenna College

Triple Non-negative Matrix Factorization Technique for Sentiment Analysis and Topic Modeling

Submitted to Dr. Blake Hunter

By Alex Waggoner

For Senior Thesis
Spring 2017
April 24th

Table of Contents

Abstract: Topic modeling refers to the process of algorithmically sorting documents into categories based on some common relationship between the documents. This common relationship between the documents is considered the "topic" of the documents. Sentiment analysis refers to the process of algorithmically sorting a document into a positive or negative category depending whether this document expresses a positive or negative opinion on its respective topic. In this paper, I consider the open problem of document classification into a topic category, as well as a sentiment category. This has a direct application to the retail industry where companies may want to scour the web in order to find documents (blogs, Amazon reviews, etc.) which both speak about their product, and give an opinion on their product (positive, negative or neutral). My solution to this problem uses a Non-negative Matrix Factorization (NMF) technique in order to determine the topic classifications of a document set, and further factors the matrix in order to discover the sentiment behind this category of product.

Introduction to Sentiment Analysis: In the United States, the incredible accessibility of the internet gives a voice to every consumer. Furthermore, internet blogs and common review sites are the first places the majority of consumers turn to when researching the pros and cons behind a product they are looking to purchase. Discovering sentiment and insights behind a company's products is hardly a new challenge, but a constantly evolving one given the complexity and sheer number of reviews buried inside a multitude of internet domains. Internet reviews, perhaps more frequently than a traditionally written and formatted magazine or newspaper review, are riddled with sarcasm, abbreviations, and slang. Simple text classification techniques based on analyzing the number of positive and negative words that occur in a document are error prone because of this. The internet requires a new solution to the trend of "reviews in 140 characters or less" which will necessitate unsupervised or semi-supervised machine learning and natural language processing techniques. Observed in [Ng et al., 2009] semi-supervised dictionary based approaches yield unsatisfactory results, with resulting lexicons of large coverage and low precision, or limited coverage and higher precision. In this paper, I will attempt to utilize these previously created dictionaries (of positive and negative words) and incorporate them into a machine learning approach to classify unlabeled documents.

Introduction to Non-negative Matrix Factorization and Topic Modeling: Non-negative Matrix Factorization has applications to many fields such as computer vision, but we are interested in the specific application to topic modeling (often referred to as document clustering). NMF is the process of factoring a matrix into (usually) two parts where a [A x B] matrix is approximated by [A x r] x [r x B] where r is a chosen value, less than A and B. Every element in [A x r] and [r x B] must be non-negative throughout this process.

We try to optimize:

$$min_{U,V} \parallel X - UV^T \parallel$$

For standard two-matrix Non-negative Matrix factorization.

There are many different ways to factor a matrix in this manner [Ho, 2008] such as multiplicative update, coordinate descent, and gradient descent methods. I implemented a custom version of the multiplicative update algorithm because it was easier to customize for my purposes rather than other, slightly more complex, algorithms.

Topic Modeling is a direct application of NMF, and attempts to summarize documents by finding common themes between them. The rank (r), in the [A x r] and [r x B] matrices can be seen as the "topic" where the a [words x documents] matrix is factored into a [words x topics] and [topic x documents] matrix. Looking at the first matrix, we can find the commonly occurring words for each topic. This would give us an approximation of a series of documents, as given by their most common topic words. We can then use the [topics x documents] matrix to backtrace the topics and find out which documents are summarized by this set of topic words.

Introduction to Linear Classification: Eventually, we will see that my algorithm attempts to classify documents based on their sentiment -- either positive or negative. Since we have labels for each data point, we can use a classifier to see how well the algorithm is performing. In this case, I chose to use a linear classifier, which separates a grouping of data points into two categories by a line. I used this to determine the ratio of the positive and negative reviews that the algorithm had correctly scored versus the amount where the algorithm failed. Assuming an equal amount of positive and negative reviews, an optimal linear classifier which got 50% of the scores correct would be equivalent to randomly guessing, where >50% would signal that the algorithm was performing better than a random guess.

Related Work: Much of the inspiration for this project was taken from a paper by [Sindhwani et al., 2009] titled *A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge*. In this paper, a possible solution to sentiment analysis upon a grouping of documents was explored. Similar to the previously described NMF-topic modeling usage, where a [words x document] matrix would be factored into [words x topics] and [topics x documents], this so called "tri-factorization" technique factors the [words x document] matrix into three matrices, [words x sentiment], [sentiment x topics], and [topics x documents]. Here, "sentiment" stands for a positive or negative value, and this [words x sentiment] matrix would have two columns, one giving a positive weight to a word, and the other column giving a negative weight to the word. Also, you may notice that [words x sentiment] · [sentiment x topics] = [words x topics]. Here you are left with the topic summary matrix as in previous sentiment analysis. However, this paper did not explore the relationship between topics and sentiment much at all, and in my paper, I was interested in exploring the relationship between [Sindhwani et al., 2009]'s sentiment analysis technique, and topic modeling through NMF. When implementing their given algorithm for tri-factorization, I found that by "recreating" the [topics x documents] matrix in this manner, I was left with some very poor results. Since the [words x sentiment] and [sentiment x topics] matrices are of rank two, the [topics x documents] matrix is also of rank two, not of rank(topics) as in standard NMF topic modeling implementations. The resultant topics are very convoluted and meaningless. In this paper, I devise a better way to determine relationships between topics and sentiment while retaining some ideas from this triple NMF sentiment analysis technique.

Background of My Technique: First, I used the total frequency - inverse document frequency (tf-idf) weighting mechanism in order to decrease the weight of commonplace words that don't bring much meaning to the document of interest. Meanwhile, words that occur in fewer documents but occur often in a handful of documents are probably important to the specific documents in which they occur, and are weighted higher. Tf-idf is a very often used technique in sentiment analysis, and is described in great detail in [Ramos].
Once applying tf-idf to the total document set, as well as removing stop-words ("the", "and", etc.) that don't supply any useful information to the document, I factored the matrix [words x documents] into [words x topics] and [topics x documents] using the Lee & Seong multiplicative update algorithm as described in [Ho, 2008]. U_0 and V_0 were initialized where each element in each matrix was assigned a value between [0, 1).

Multiplicative Update Algorithm (Dual Factorization): $X \approx U \cdot V^T$, find $min_{U,V} \parallel X - UV^T \parallel$

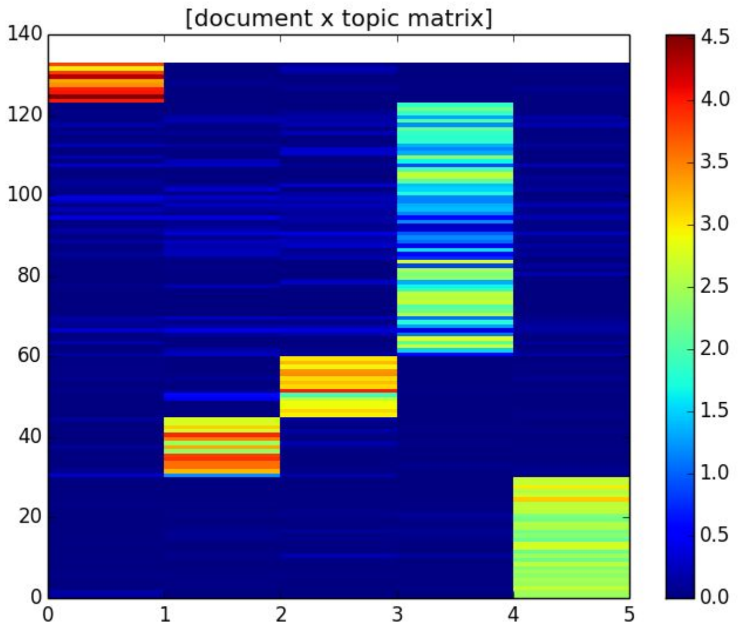
```
Step 1: Randomly initialize $U_0$, $V_0$, set k=0
Step 2: **while k<iterations, loop:**
Step 3:     $U_{k+1} = U_k \circ \frac{X \cdot V_k}{U_k \cdot (V_k)^T \cdot V_k}$
Step 4:     $V_{k+1} = V_k \circ \frac{X^T \cdot U_{k+1}}{V_k \cdot (U_{k+1})^T \cdot U_{k+1}}$
Step 5:     k = k + 1
Step 6: **end while loop**
```

This factorization can be used to pull out the most heavily weighted words in the topics by creating a mapping from the values of the elements in the columns of U back to the actual words these values represent. Then by sorting the mapped (word, value) pairs from greatest to least, we can then take the greatest 20 values in order to receive a concise description of each topic. With a sample document set where there were 5 types of documents: baseball, dinosaurs, World War II, finance, and philosophy, here are 5 topics from this double NMF algorithm:

```
Topic 1 :
invasion allied troops normandy german british germans landing france beaches 000 eisenhower divisions beach operation allies 1944
june day

Topic 2 :
dinosaurs dinosaur dr scientists bones said years fossils species university museum ago eating animals long animal birds fossil mr

Topic 3 :
game hit run said runs innings jays twins inning blue season toronto left right reds games hits home homer

Topic 4 :
percent quarter said billion revenue fund year company share sales shares million hedge cents bank funds market growth investors

Topic 5 :
truth opinions reason certain men true objects thought mind god nature order thoughts knowledge ought sciences doubt sufficient
principles
```

Without a mathematical proof, one can imagine that each of the five types of documents is heavily correlated to its respective topic. Topic 1 is clearly about World War II, more specifically the battle of Normandy (D-Day) which took place in 1944. Topic 2 is clearly about dinosaurs, fossils, museums … Topic 3 is clearly about baseball, perhaps more specifically the Toronto Blue Jays, Minnesota Twins, and the Cincinnati Reds. Topic 4 is about finance, economics and banking. Finally, topic 5 is about philosophy topics: truth, opinions, man, nature and god. Clearly, our topics are very coherent and effectively summarize the documents given. Even specifics to the document set (World War II, but specifically D-Day, etc.) can be seen in our topic analysis.

We can now visualize the matrix V and show that each document corresponds heavily to one topic, and little to no correspondence to any other topic.



The topics look so well grouped because the documents were paired together (in the same file system folder) when input into this algorithm. What I do want to draw your attention to, is the fact that each document (row) has one strong coloration, and four weak (blue, light blue) colorations. As previously stated, each document is highly correlated to **one** topic only. This will become important later, where we attempt to assign a sentiment value to one grouping of topics.

My Technique: I seek to get an overall sentiment for each topic category. I define sentiment as solely positive or negative sentiment. Other categories of sentiment are unexplored in this paper. We can expand the standard $X \approx U \cdot V^T$ factorization scheme into $X{\approx}F \cdot S \cdot V^T$ where F is a [words x sentiment categories] matrix, and S is a [sentiment categories x topics] matrix, where all matrices U, V, F and S are non-negative. (Note that sentiment categories equals 2, as explained above.) Therefore, $U = F \cdot S$. Theoretically, the U obtained from our initial, dual matrix factorization scheme should be equal to $F \cdot S$ in our triple matrix factorization scheme, however experimentally, this is not the case. Since we are only using two sentiment categories, matrices F and S^T are of rank 2. Therefore $F \cdot S$ are also of rank 2, while our topic matrix given by dual factorization is of rank r. Therefore, my solution was to find U in the usual way by dual factorization, and then triple factor the matrix while holding the topics constant.

Expand $X \approx U \cdot V^T$ into $X{\approx}F \cdot S \cdot V^T$ where $U = F \cdot S$ for tri-factorization.

This was done by finding V in the dual factorization algorithm, and then using this same V (setting it equal to V_0 for stylistic reasons below) in the triple factorization approach. As shown above, V holds references to which documents the topics correspond to. By holding V constant in our algorithm, we maintain these references. Now we can look at S, the [sentiment x topics] matrix and be assured that we are receiving a sentiment value for coherent topics.

We initialize the matrix S as randomly as before where each element in the matrix is assigned some decimal value in [0, 1). However, F, as the [words x sentiment classes] matrix, we seed this matrix with some previous lexical knowledge. Thanks to [aaa], I had a set of ~2k positively correlated words, and another ~2k negatively correlated words. To initialize the F matrix, instead of purely randomly as done previously, we used this lexical knowledge. If word, *w*, was in the set of positive words then $(F_0)_{w0} = 1$. Oppositely, if word *w* was in the set of negative words then $(F_0)_{w1} = 1$. All other elements in the matrix, which were not labeled in the lexical knowledge, are randomized as before where each element is assigned some decimal value in [0, 1).

As in the dual NMF implementation, where we algorithmically solve $min_{U,V} \parallel X - UV^T \parallel$, here we try and solve $min_{F,S} \parallel X - FSV^T \parallel$, where all matrices U, V, F and S are non-negative. Notice that X is fixed in both equations, and V^T is determined in the dual NMF equation and then fixed in the triple NMF equation as explained previously. However, we will make a slight improvement upon this triple factorization minimization function and add a penalty function in attempt to keep the structure of F_0, our [words x sentiment classes] matrix, w.r.t. the “known” words from our previous lexical knowledge. With this in mind, we will try and minimize

$$min_{F,S} \parallel X - FSV^T \parallel + \alpha Tr[(F - F_0)^T \cdot \psi \cdot (F - F_0)]$$

where Tr(A) stands for the trace of matrix A. ψ is a function where

$$\psi = \begin{cases} 1 & when\ word \in lexicon \\ 0 & when\ word\ unknown \end{cases}$$

When $\psi = 1$, we know that the specific word belongs to either the positive or negative word set, and we try and force F to honor this. While if $\psi = 0$, we do not care about the form of F, because we do not have prior knowledge about this word. It may either have positive, negative, or neutral sentiment. The parameter $\alpha > 0$ represents the amount we want to force $F \approx F_0$. Usually, in testing, we kept α small because this prior lexical knowledge was curated for general purposes, and not specifically for our test data sets (e.g. Amazon reviews). By keeping α small, we form a semi-supervised learning algorithm. A larger α would make a supervised learning algorithm, and setting $\alpha = 0$ would give an unsupervised algorithm.

Double Multiplicative Update Algorithm: ($X \approx U \cdot V^T \rightarrow X \approx F \cdot S \cdot V^T$)
Step 1: Randomly initialize U_0, V_0, set k = 0

```
Step 2: while k<iterations, loop:
Step 3:     $U_{k+1} = U_k \circ \frac{X \cdot V_k}{U_k \cdot (V_k)^T \cdot V_k}$
Step 4:     $V_{k+1} = V_k \circ \frac{X^T \cdot U_{k+1}}{V_k \cdot (U_{k+1})^T \cdot U_{k+1}}$
Step 5:     k = k + 1
Step 6: end while loop
Step 7: Set $F_0$ as described, randomly initialize $S_0$, fix $V_k$ as $V_0$, set k = 0
Step 8: while k<iterations, loop:
Step 8:     $S_{k+1} = S_k \circ \frac{F^T \cdot X \cdot V_0}{F^T \cdot F \cdot S \cdot (V_0)^T \cdot V_0}$
Step 9:     $F_{k+1} = F_k \circ \frac{X \cdot V_0 \cdot (S_{k+1})^T + \alpha\psi F_0}{F_k \cdot (F_k)^T \cdot X \cdot V_0 \cdot (S_{k+1})^T + \alpha\psi F_k}$
Step 10:    k = k + 1
Step 11: end while loop
```

A proof that this algorithm converges to a local minimum is given in [Sindhwani et al., 2009].

Results: To test this new algorithm, I needed to use a set of document where a distinct topic could be found amongst groups of documents, and a sentiment value could be calculated amongst all these documents. A grouping of Amazon reviews would work just fine, where I would pull multiple different items, and a grouping of their respective reviews to analyze topic (which items were being reviewed?) and sentiment (what did reviewer's score these items, overall?).

Dataset 1: 150 Amazon reviews, 5 given categories, 5 topics, 2 sentiment classes, α = 0.1
The first dataset consisted of 30 Amazon reviews of various inter-domain items with low, neutral, and high scoring reviews. The first was a book titled *Org's Odyssey* with **very** low reviews. Every review was one star (except one), and review authors had only negative things to say about this book. Secondly, I picked a video game titled *SimCity: Limited Edition* which also had very low ratings. However, there were (few) good things about this game which reviewers pointed out. Overall, this game averaged 1.5 stars. Next, was a type of chapstick with incredibly high reviews called *Natural Ice Lip Balm*. This product has a solid 5.0 stars out of 500+ Amazon reviews, and there were very few negative things said about this product. Next was a product called *Light My Fire Swedish FireSteel*, which was an emergency tool to light fires. This

received high reviews as well, and averages 4.5 stars on Amazon. Next was *Bullshooter by Arachnid Electronic Dartboard Cabinet Set* which was neutrally rated on Amazon at 3.0 stars.

I ran the algorithm on this dataset with $\alpha = 0.1$. This is a fairly low, but still significant value. Since the sentiment lexicon was not curated for Amazon reviews, and since the reviewed products are cross domain where each product has little in common with other products, a less-supervised approach seems reasonable. A better related sentiment lexicon, and single-domain reviews can use a greater valued α and will perhaps give better results. In this way, this algorithm is quite customizable.

Here is the printout from the algorithm:

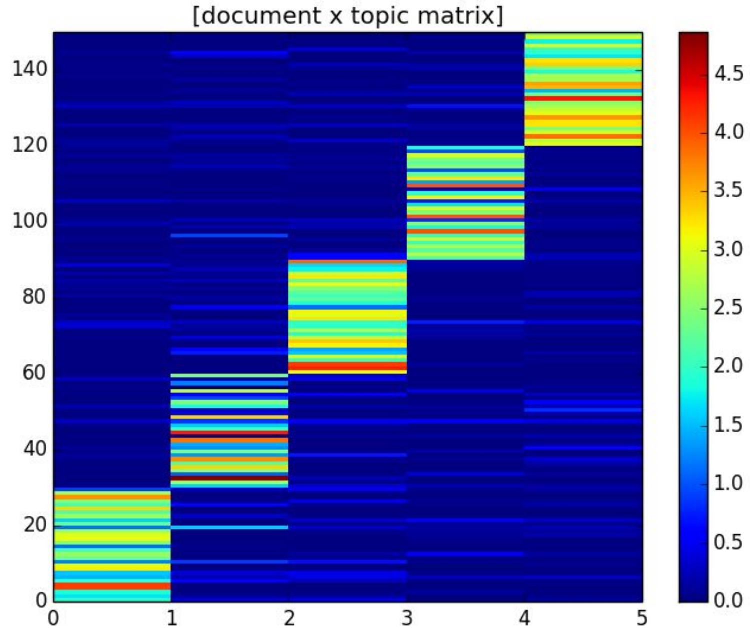
```
Topic 1 :
darts board stopped sound heckler bought set fun came buy love product return stick bounce recommend away good stay
positive correlation: 0.0891465825713
negative correlation: 0.00996225037894
positive/negative: 8.94843827252

Topic 2 :
tinder firesteel sparks cotton survival use light striker lint magnesium model need good balls great paper flint whistle work
positive correlation: 0.161486888709
negative correlation: 0.00371155370457
positive/negative: 43.5092421026

Topic 3 :
lip balm lips natural used ice best years brand chapstick like chapped dry ve moist doesn flavor effective fast
positive correlation: 0.110901294239
negative correlation: 0.000273794658986
positive/negative: 405.052803621

Topic 4 :
game play city ea simcity servers cities just games issues online hours server save want build player work playing
positive correlation: 0.0283524973866
negative correlation: 0.0686796275023
positive/negative: 0.412822527112

Topic 5 :
book read worst furry say finish literally manage dump ear diarrhea buffalo thing written awful couldn author like wish
positive correlation: 2.3028448365e-06
negative correlation: 0.100780489907
positive/negative: 2.28501055972e-05
```

When looking at the positive/negative correlation ratio, this algorithm worked perfectly in determining the ranking of the different products. From worst to best, the algorithm ranked:

Product	Algorithmic ranking	Amazon ranking (stars)

Org's Odyssey - topic 5	0.0000228501055	1.0
SimCity: Limited - topic 4	0.412822527112	1.5
Electronic Dart Board - topic 1	8.94843827252	3.0
Emergency Fire Starter - topic 2	43.5092421026	4.5
Natural Ice Lip Balm - topic 3	405.052803621	5.0

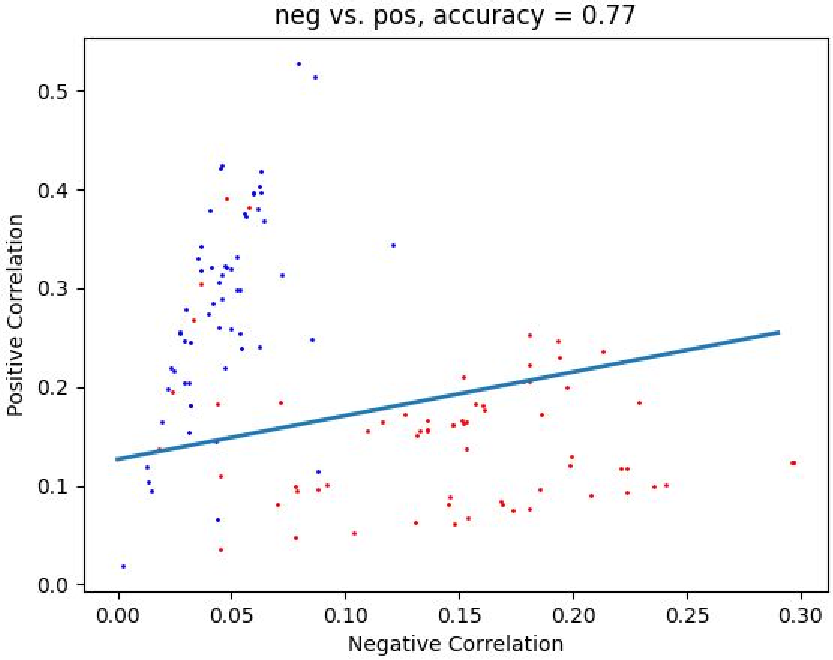
As before, with the sample NMF dataset, each document is mostly correlated to a single topic. This is important to verify my claim that this algorithmic sentiment rating for a product is correlated the Amazon stars ranking for that same product. Here is a visualization of the [document x topic] matrix V.



Clearly, the algorithm performed as expected with each product being ranked by the algorithm in the same order as their given "star" count. However, the algorithmic ranking varies greatly from the given star count, but I would argue that results like this are to be expected. Some products are so terrible that reviewers often state "I wish I could give zero stars" or even "negative stars". While some products are so incredible that all reviews are extremely positive. Clearly, when looking at the actual Amazon reviews, and my algorithm's output, the *Natural Ice Lip Balm* was one of these products, and is among the highest ever reviewed products on Amazon. Likewise, *Org's Odyssey* is supposed be one of the worst products ever reviewed on

Amazon, and this was also reflected in my algorithm. These outputs can be seen as an experimental lower and upper bound to this algorithmic ranking system. To transform the algorithmic rankings back to Amazon 1-, 2-, 3-, 4-, or 5-star rankings, we would need some type of normalization function to scale reviews to their respective star ranking.

Next, I looked at the individual document sentiment rankings. Since I was using Amazon reviews, I had the star ratings of each review to easily test how well my algorithm was performing. Taking the matrix $D = S \cdot V^T$, we had a matrix of the form [document x sentiment]. We can plot the point by their positive vs. negative sentiment scores, and then find a linear classification line to try and predict which documents belong to which categories. In a scatter plot shown below, we can see that it's quite easy to separate these two categories of points, and a linear classifier will work well.

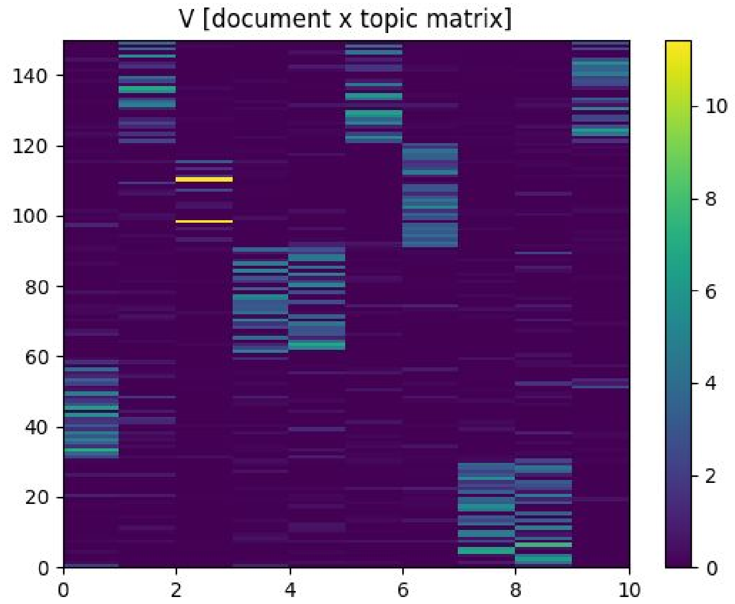


The 5-star amazon reviews from the dataset are colored blue, while the 1-star are colored red.

An optimal linear classifier will be correct 77% of the time with this dataset (shown in the title of the graph).

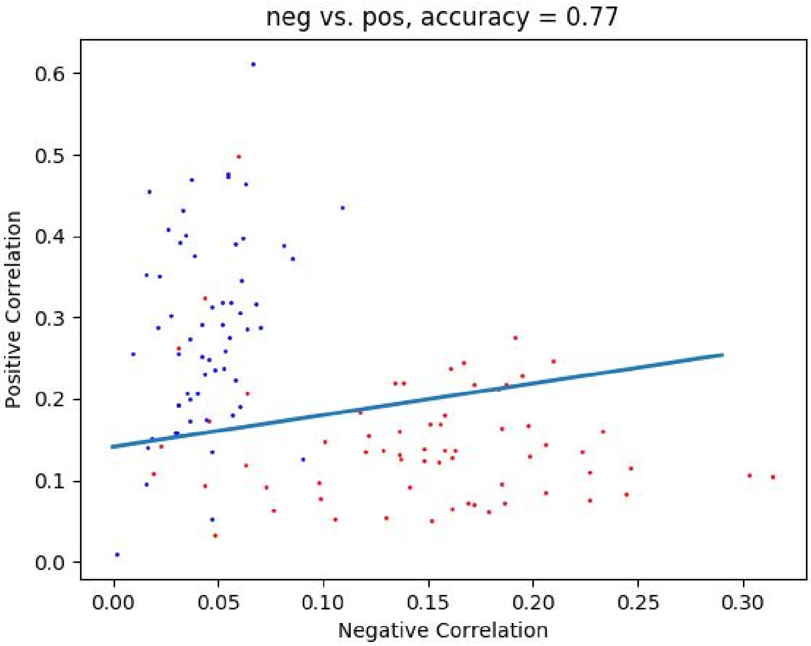
Dataset 2: 150 Amazon reviews, 5 given categories, 10 topics, 2 sentiment classes, α = 0.1

For the next test, I ran the same dataset, but changed the parameters to see if we could get better scoring. This time, I raised the amount of topics to 10, rather than 5 before. Since there are 5 products, the topics won't correspond 1:1 with each product, but we shouldn't expect this to occur with every dataset. Rather, topic modelling should be viewed as a process to find topics common to all, or subsets of the documents. Since, in the 150 review dataset, the products are all fairly disjoint, we had this nice distribution before, but with larger datasets, we won't be sure that all categories of documents are disjoint, and they usually will not be.



Here we see the document x topic matrix for this dataset 2: 150 documents, 10 topics, and 2 sentiment classes.

Even though there are now twice as many topics, we can see that each topics describes documents that are fairly grouped together, signifying that each topic describes one topic.

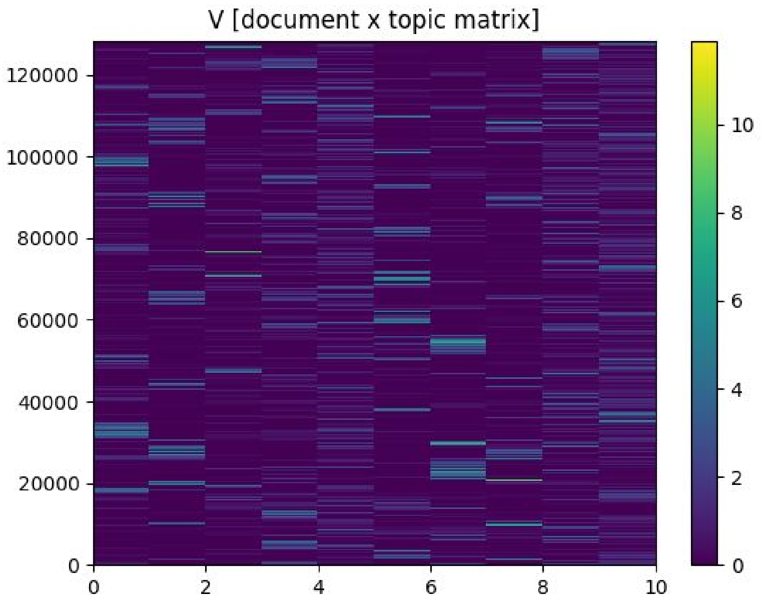


Here we see the per-document accuracy for this same run. The accuracy hasn't changed at all when adding the extra topics.

In this run, we found that with this small dataset, varying the number of topics didn't change the accuracy ranking per document, and didn't bring much of a noticeable change to our algorithm,

Dataset 3: 128400 Amazon reviews, 100 given categories, 10 topics, 2 sentiment classes, α = 0.1

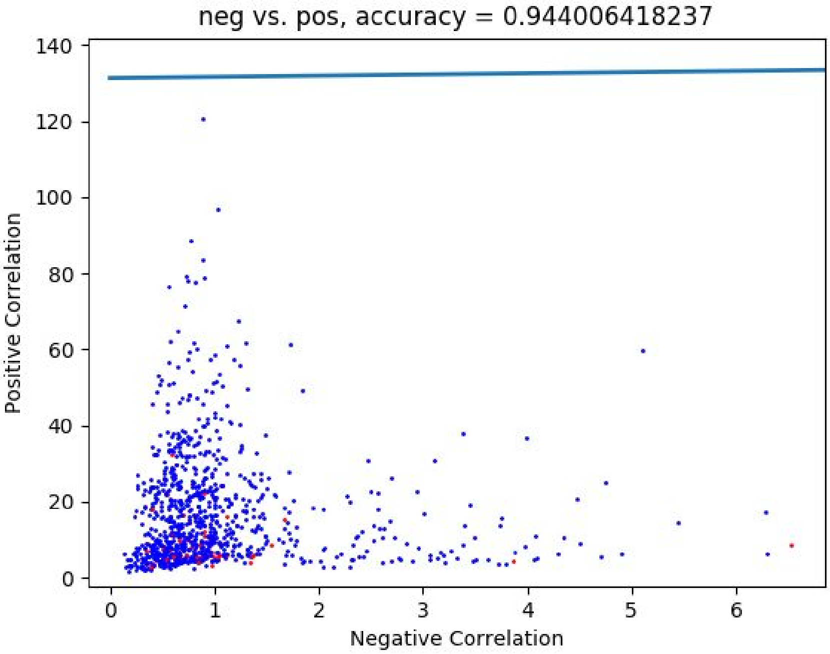
Next, I ran a much larger dataset though my algorithm, to try and test the limits of what it can and cannot do. I used a subset of the Amazon Electronics dataset from [123] to run my tests. I pulled out the 100 most reviewed products in this dataset, and ran them through my algorithm. Since all these products belong to the "electronics" category of Amazon reviews, there is a lot of overlap between the products being reviewed (multiple TV's, computers, smartphones, etc.)



Here is the topic document x topic matrix for this run. We see that some of the topics describe only a few select products, while some topics slightly describe all products, or most products.

```
Topic 1 :
tv roku apple netflix device watch use chromecast streaming 34 remote channels video movies youtube stream mount apps google
positive correlation: 0.886720742549
negative correlation: 0.0483707978365
positive/negative: 18.3317369613

Topic 2 :
cable hdmi quality apple works length does picture just high needed need lightning connect connectors box say amazon sturdy
positive correlation: 0.669337379289
negative correlation: 0.104966601599
positive/negative: 6.37666999876

Topic 3 :
mouse keyboard use logitech battery batteries wireless used buttons laptop small receiver hand keys like using trackball button life
positive correlation: 1.07837141659
negative correlation: 0.0796869929607
positive/negative: 13.5325901571

Topic 4 :
kindle charger charge usb phone charging car power plug use charges ipad iphone cord like ports devices cover device
positive correlation: 1.17522042466
negative correlation: 0.0466009986054
positive/negative: 25.2187819968

Topic 5 :
great works product price recommend buy say bought problems perfectly fast love easy item fine issues highly use advertised
```

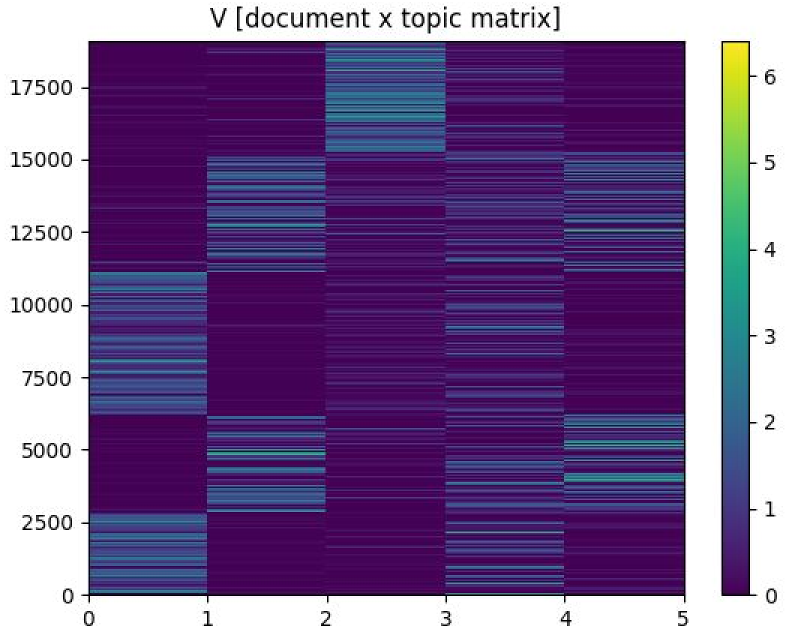
```
positive correlation: 15.0855555437
negative correlation: 0.108419834821
positive/negative: 139.140181947

Topic 6 :
drive hard drives usb flash external backup files seagate data wd computer storage fast ssd small windows use software
positive correlation: 0.762783868277
negative correlation: 0.774192644252
positive/negative: 0.985263647156

Topic 7 :
card memory camera sd cards sandisk class fast video pictures phone gb 10 speed canon use photos storage bought
positive correlation: 0.816980564032
negative correlation: 0.143760981197
positive/negative: 5.68290893141

Topic 8 :
cables hdmi work quality amazon buy expensive price high mediabridge money don monster ve cheap bought need just best
positive correlation: 0.774930985975
negative correlation: 0.163716842775
positive/negative: 4.73336141133

Topic 9 :
good quality sound price ear headphones product fit really buds better like bass buy earbuds ears nice comfortable pair
positive correlation: 4.0941001621
negative correlation: 0.154865697485
positive/negative: 26.4364557716

Topic 10 :
router just modem easy wireless time like 34 set work ve got don old did internet new signal setup
positive correlation: 1.78711874573
negative correlation: 0.318446140921
positive/negative: 5.61199686879
```



Here, we can see that this dataset ran into problems while attempting to classify on a per-document basis. An optimal linear classifier decides to classify all points as a single class. This makes sense when we look at the layout of the Amazon of this dataset. We find that 94.4% of the reviews are five-star reviews compared to one-star reviews. To beat a linear classifier, my algorithm would have to perform at a accuracy rating of >94.4%. As we saw in the previous dataset, this wasn’t the case, and my algorithm performed at ~75% accuracy with a review set where the number of five-star vs. one-star reviews was approximately equal. The algorithm cannot cope with this massive amount of positive reviews.

Dataset 4: 19104 Amazon reviews, 5 given categories, 5 topics, 2 sentiment classes, α = 0.1

This dataset consisted of the top 5 Amazon products classified as electronics.



```
Topic 1 :
card memory sd camera cards class phone sandisk fast 10 galaxy pictures storage speed gb bought use video micro
positive correlation: 0.755207951025
negative correlation: 0.0913396960688
positive/negative: 8.26812419494

Topic 2 :
cable hdmi quality tv good picture high does just needed connect length need say player box ray 3d price
positive correlation: 0.761114813212
negative correlation: 0.0559660748165
positive/negative: 13.5995746657

Topic 3 :
tv chromecast device netflix google use youtube apps stream chrome streaming phone 34 easy just watch video roku app
positive correlation: 0.817918706289
negative correlation: 0.0338391477617
positive/negative: 24.1707832611

Topic 4 :
great works price good product recommend quality say buy problems fast bought use issues item beat highly perfectly camera
positive correlation: 7.92461300151
negative correlation: 0.0643041172524
positive/negative: 123.236479095

Topic 5 :
cables hdmi quality work buy mediabridge amazon expensive money high don monster price ve just best purchased bought brand
positive correlation: 0.649041875964
negative correlation: 0.114953795184
```

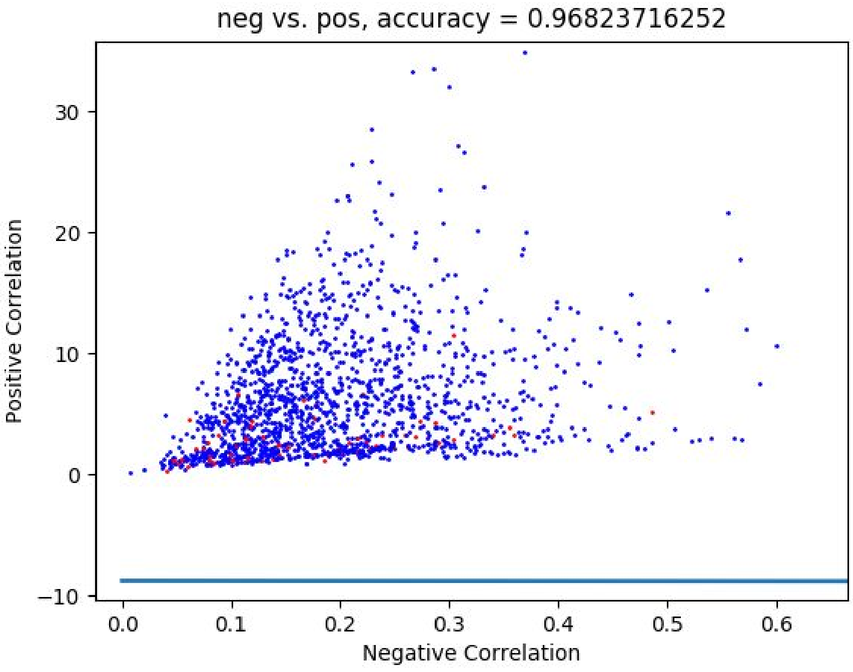
```
positive/negative: 5.64611089983
```

Since we have less product categories in this dataset, the topic modelling portion was much easier to visualize. The Amazon products being reviewed are below:

Product 1: Sandisk 64GB MicroSD card
Product 2: Amazon high speed HDMI cord with ethernet
Product 3: Power cord adapter for Google Chromecast
Product 4: Transcend 10GB SD card
Product 5: Mediabridge 6 ft. HDMI Cable

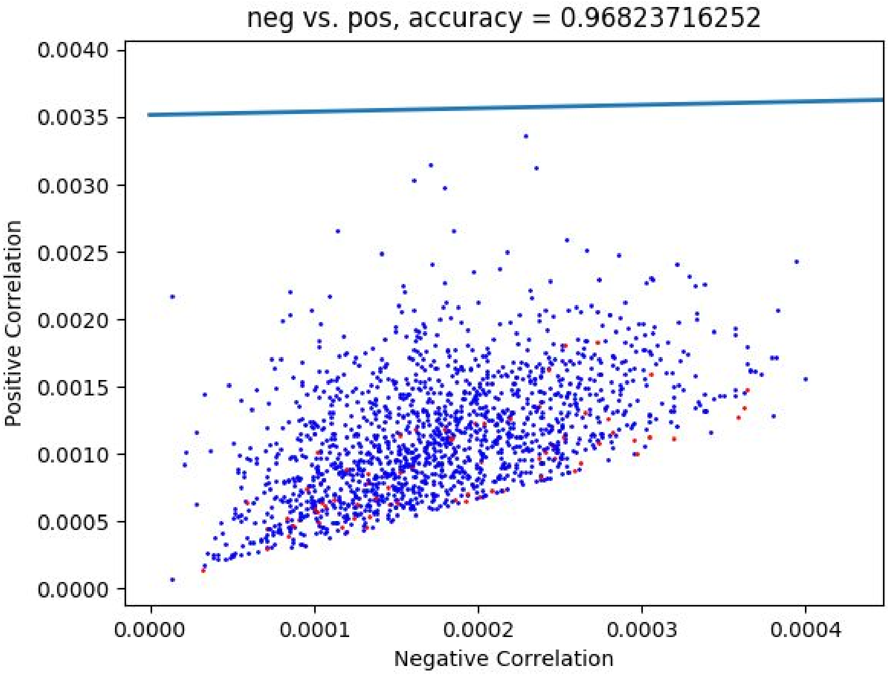
We can see that the topics accurately describe these products. The positive/negative correlation is high for each product, which makes sense given the earlier problem that the vast majority reviews are 5-star reviews.

To reinforce this point, we can look at topic 4 which is fairly evenly distributed amongst all products. This topic pulled out all extremely positive words from the documents (ex: *great, works, price, good, product, recommend, quality, etc.*)



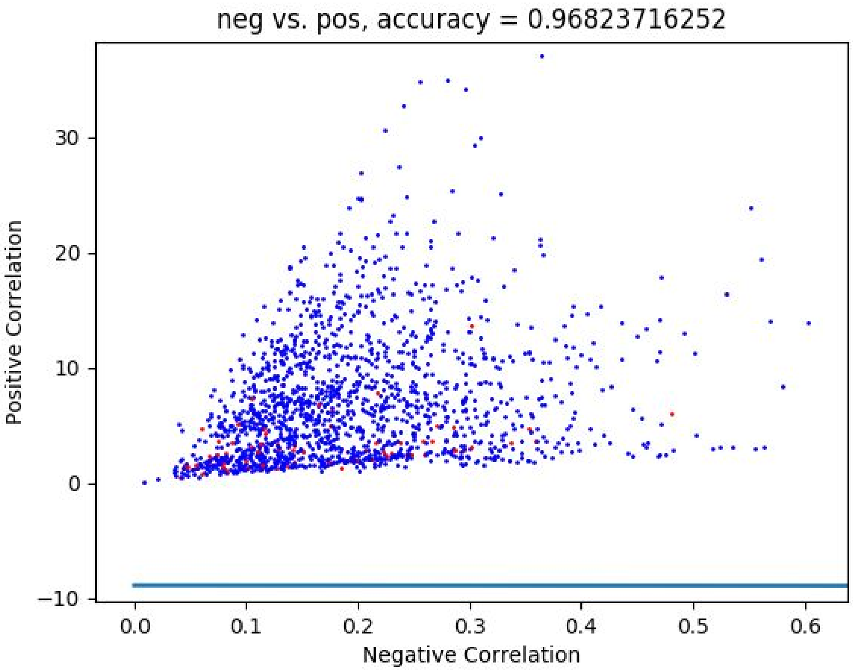
Again, we have the problem where per-document sentiment analysis is not performing as expected, with all reviews being categorized similarly. I attribute this to the same problem as before, where all these products are very highly reviewed on Amazon.

Dataset 5: 19104 Amazon reviews, 5 given categories, 5 topics, 2 sentiment classes, α = 10.0
Here, I used an identical dataset as before, but changed the parameter **α** in my code from **α**=0.1 to **α**=10.0 in effort to force the scatterplot of the predictive sentiment accuracy to be linearly separable. As before, the topic distribution is the same, and sadly, the accuracy ranking changed little too.



As you can see, again, this scatterplot is non-linearly separable, and the accuracy ranking is unchanged. The linear classifier categorizes all reviews into one category without distinction.

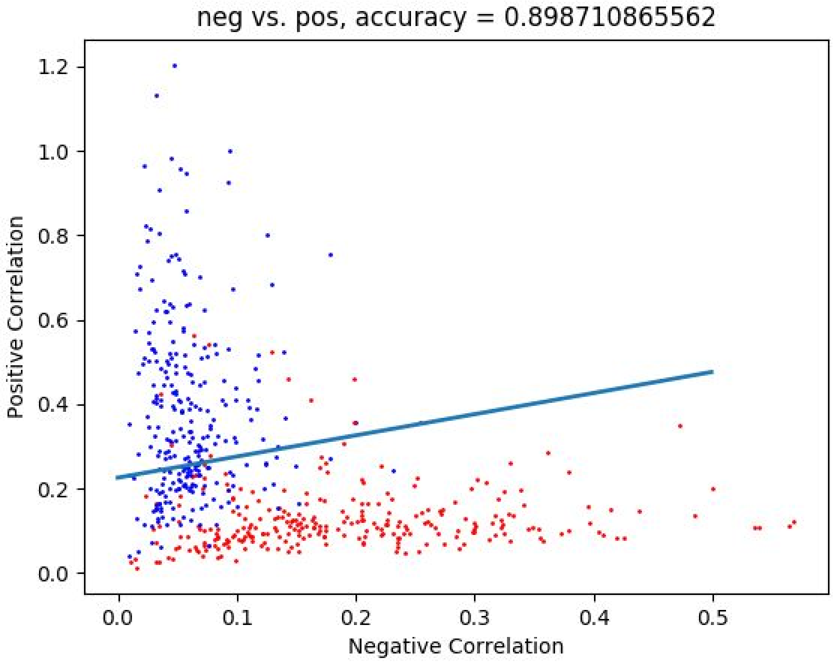
Dataset 6: 19104 Amazon reviews, 5 given categories, 5 topics, 5 sentiment classes, α = 0.1
Again, we used the same dataset, but changed the dimensions of the sentiment matrix in the triple factorization step. This was an attempt to allow clearly positive and negative words to be placed into the first two rows of this matrix as before. However, some words in the English language do not have any overt positive or negative correlation. By adding three more rows, I hoped that the words that are neither negative nor positive would “trickle down” and we would have better accuracy results when these words did not interfere with the positive or negative ranking. However, as we can see below, this was not the case, and the same problems occurred once again.



All data points were classified the same, because the dataset was non-linearly separable as before.

Dataset 7: 544 Amazon reviews, 2 given categories, 5 topics, 2 sentiment classes, α = 0.1
Finally, I manipulated the first two products of the Amazon dataset, so that the total amount of 5 star reviews equalled the total amount of 1 star reviews. I wanted to test the hypothesis that the per document sentiment analysis was failing because of the mismatch between positive and negative reviews. Looking at the figure below, the problem from before (non-linear separability) was fixed by equalizing this ratio.

Clearly, a limitation of this algorithm is that the amount of positive vs. negative reviews must be similar. In retrospect, this makes sense, because this is a comparative algorithm which compares documents against the whole to determine their sentiment. If almost all reviews are positive, the algorithm ends up comparing positive reviews against the average baseline (which is already extremely positive), and the algorithm fails. However, is the average baseline is neutral (which it will be when there is an even spread between positivity and negativity), then the algorithm works as expected, where positive reviews are ranked above the average review sentiment, and negative reviews are ranked oppositely.



Conclusions: This algorithm performed well in terms of topic modelling and topic sentiment analysis. The algorithm easily picked up common themes amongst a corpus of documents, as well as the sentiment of these themes. However, it performed poorly in terms of per-document sentiment analysis when there was a large amount of positive reviews compared to negative reviews. When there was a similar ratio of positive to negative reviews, the algorithm performed fairly well in per-document sentiment analysis, with accuracy scores >75%. This seems consistent with the accuracy scores found in [Sindhwani, 2009] using a similar algorithm. However, my algorithm has the added benefit of increased attention paid to topic modelling, instead of a purely sentiment analysis driven algorithm.

Future work would contain more testing of sentiment balanced datasets, as well as supervision on the document side. Currently, this algorithm is partially-supervised on the lexical side, and un-supervised on the document-sentiment side, as well as in topic modelling (which is typically an unsupervised process). With added supervision, better results would be expected as shown in [Sindhwani, 2009].

References:

[Ho, 2008] Ho, N.-D. (2008). *Nonnegative Matrix Factorization Algorithms and Applications.* PhD Thesis, Universit Catholique De Louvain.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). *Learning the parts of objects by non-negative matrix factorization.* IEEE Signal Processing Letters, 401(6755):788–791.

[Lin, 2007] Lin, C.-J. (2007). *On the convergence of multiplicative update algorithms for non-negative matrix factorization*. IEEE Transactions on Neural Networks, 18(6).

[Sindhwani, 2009] Sindhwani, V., Zhang, Y., and Li, T. (2009). *A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge*. IBM T.J. Watson Research Center.

[Ng et al., 2009] Dasgupta, S. and Ng, V. (2009). *Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification.* ACL- IJCNLP: Volume 2, pages 701–709. Association for Computational Linguistics.

[Ramos] Ramos, J. *Using TF-IDF to Determine Word Relevance in Document Queries.* Rutgers University.

Annett, M. and Kondrak, G. *A Comparison of Sentiment Analysis Techniques: Polarizing Movie Blogs*. Department of Computer Science, University of Alberta.

Flenner J. and Hunter, B. (2017). *A deep nonnegative matrix factorization*. PhD Thesis in Mathematics, Claremont Graduate University.

Datasets:

Liu, Bing, and Minqing Hu. "Opinion Lexicon (or Sentiment Lexicon)." UIC Computer Science. N.p., 15 May 2004. Web. 06 Apr. 2017.

McAuley, Julian. "Amazon Product Data." Amazon Review Data. UC San Diego, n.d. Web. 06 Apr. 2017.

Python Libraries:

Scikit Learn. Computer software. Scikit-learn Machine Learning in Python. Vers. 0.18.1. N.p., 2010. Web.

Matplotlib. Computer software. Matplotlib. Vers. 2.0.0. NumFOCUS, 2002. Web.

NumPy. Computer software. Numpy. Vers. 1.12.1. NumFOCUS, 2005. Web.

Bicking, Ian. Virtualenv. Computer software. Virtualenv. Vers. 15.1.0. The Open Planning Project, PyPA., 2007. Web.