DV2540 – Machine Learning

Project – Describing movies using Topic Modeling

Mattias Hjalmarsson
890604-3552

I. Introduction

The aim of this project is to provide a system which automatically describes movies (by finding topics) using movie summaries, with the help of topic models. *In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents [1].*

Topic models can be used in different areas for different problems. The most common problems solved today, with the help of topic models, are content-based recommendation systems, data representation and exploration/searching. In regards to the movie domain, not that much previous research exists. See "Related work" for more information.

The contribution of this project is two-fold. It could be used to automatically provide keyword descriptions of movies, in turn enabling users to easily overview the topical structure of a movie. This could then be used in different movie database sites to provide a quick overview. Besides description, it could also be used as a base to further build a recommendation system upon. For example, by providing recommendations based on similarity of different topic distributions in movies.

The expected results are that the model will provide understandable, coherent topics, and that it will be able to infer the latent topics of a few test cases using existing summaries for these movies.

II. Related Work

As mentioned in the introduction, not much research in the movie domain has been conducted. However, some papers exist.

One paper presented a comparison between topic models for a movie recommendation system [2]. They compared Latent Dirichlet Allocation (LDA) and Latent Semantic Allocation (LSA) also called Latent Semantic Indexing (LSI). Their intention was to test these two different topic models in a movie recommendation system. Their work could be considered to capture a lot of what is to be done in this project.

Another paper presented a similar system where they would use both user ratings and movie plot similarities to recommend movies [3]. The plot similarities were calculated with the help of LDA.

The goals of these papers differ from the goal of this project. The output from their models are not observed and used directly by humans, but rather used in providing recommendations of other similar movies. Thus, their models do not have to be as human readable and semantically correct in topic assignments.

III. Method

The project will be executed in different stages. The first stage includes finding a usable dataset which can be used to *train* the model. Training in this case, means fitting the model with data that enables it to later infer latent topics in movies not existing within the training dataset. The second step involves pre-processing the selected dataset. This stage is arguably the most important one, since the topics learnt by the model are dependent on the training data. This means that the quality of the input data affects the quality of the output data, *garbage in, garbage out.* The third stage is deciding on the topic model that is to be used. The last stage concerns evaluation and parameter tweaking to get the best possible result out of the model.

A. Training

Deciding on a dataset to fit the model is an important step. There are a few factors that need to be considered. The dataset needs to represent the domain in which it will be used. For example, having data regarding only biology will most likely result in topics only concerning biology. Thus, this project requires a largely scoped dataset. Finding one in this domain is tricky, since most research concerning movies uses meta-data (ratings, genres, reviews etc.) and not "content specific" data like summaries and synopses. The result of this is that most open/free datasets in the movie domain concerns meta-data. Two different datasets were found and considered for this project.

CMU Movie Summary corpus contains, amongst other data, 42,306 movie summaries extracted from Wikipedia [4].

OPUS the open parallel corpus, which contains subtitles which in total contains 2,6 billion sentences across 60 languages [5].

After both had been pre-processed (pre-processing explained under "Pre-processing") the datasets were evaluated. CMU was decided to be the most fit dataset for this problem. This is since even though subtitles contain more words than summaries, they contain less descriptive words, more names and more stopwords (stopwords explained under "Stopwords"). The OPUS dataset seemed to include a lot of TV-series with many different episodes. This resulted in topics being very focused on names appearing together, and not descriptive nouns. Since OPUS contains mostly user-made subtitles (often not run through quality checks), many spelling errors, missing spaces, and more grammatical errors were found. This resulted in lower quality data. CMU also contains user-made data, however it could be less prone to errors because of multiple people reading and quality checking the content on Wikipedia. As well as summaries being smaller in length and therefore easier to error-correct.

B. *Pre-processing*

As noted earlier, pre-processing of data is crucial. Thus, much time and effort were put into the different pre-processing steps. What each step consists of and why it is needed will be explained below. Note that *summaries* and *documents* will be used interchangeably to describe the data.

1) *Tokenization*

To enable the coming stages of pre-processing, the CMU dataset had to be loaded and tokenized. In this project, tokenizing words means to split the summaries into words, called tokens. Every summary in the CMU dataset exists on a separate line, this means that reading line by line results in each summary being read one after another. Each line (summary) can then be tokenized for further processing. A summary could normally contain several separated paragraphs. Unfortunately, the CMU dataset groups all the paragraphs of the summary together into one long string. Since different paragraphs can contain different themes, grouping them will remove this separation and create more of a generalized thematic structure. This issue could somewhat be counteracted by using a technique called *chunking* explained under “Improvements & future work”.

2) *Stopwords*

Stopwords are words that are very common in a language and provide no real description or meaning. For example, words like, *his*, *is*, *at* and *the* are considered stopwords. There exist no universal stopword list, but rather each list should be tailored to the specific needs of the user. In this project, stopwords consist of not only common words, but common names as well. The movie domain contains a lot of names. These needs to be filtered out, otherwise the resulting topics will be full of names that provide no real meaning to the topic itself. This is a difficult task considering the number of names that exist. Fortunately, a stopword list containing (besides common words) around 5600 common names were found and used as a base list for this project [6]. This list was continuously appended to throughout the project, as common words/names were found.

3) *Part-of-Speech tagging*

Part-of-Speech tagging *...is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph [7]*. Words can have more than one meaning, e.g. date, leaves and type. Being able to figure out whether a word is a verb (he *leaves*) or a noun (many *leaves*) is of great value when trying to find words of a thematic nature. Nouns provide more value than for example adjectives and verbs in this project. Therefore, filtering out all words that are not nouns results in more thematic topics. Proper nouns specify names of persons and things. Removing proper nouns and only keeping common nouns results in many names being removed. This further improves the quality of the training data. Unfortunately, names of things could provide much descriptive power but getting rid of names is of higher priority. Part-of-Speech tagging will be implemented using the *Perceptron tagger* from the *nltk* (Natural Language Toolkit) python library [8].

4) *Lemmatization and Stemming*

Lemmatization concerns finding the *lemma* of a word. A lemma is the canonical form of a word, i.e. give, giving, given are all forms of give, meaning that give is the lemma. Besides finding the lemma of a word, the lemmatization technique will find the singular form of a word, i.e. *apples* will be changed to *apple*. With the help of part-of-speech tagging, lemmatization techniques can predict the correct type of lemma, meaning it knows for example whether a word is a verb or noun and can therefore find the correct lemma. After lemmatization, the words can be grouped together (*give*, *given*, *giving* will all collected in *give*), massively reducing the number of words (features).

Stemming is another technique for finding the *stem* of a word. Stemming will cut words to their stem, as an example the word *apples* would be cut to *apple* and *giving* would be cut to *giv*. As seen, the result could be non-existing words. Since the words are to be read by humans, such non-existing words are not wanted. Therefore, lemmatization is a technique more fit for this problem. Lemmatization will be implemented using the *WordNetLemmatizer* from the *nltk* python library [9].

5) *Frequency removal*

Even though stopwords are removed, not all common words can be found and addressed. By removing words that occur too frequently as well as remove words that only exist in a few summaries, the number of words (features) can be reduced even more. As mentioned earlier, the quality of the input data is important.

C. *Deciding on a topic model*

There exist multiple different topic models, the most researched and used topic models are Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA) where LDA is often what people refer too when the speak of topic modeling [10][11]. In this project, a quick comparison between LSI and LDA was conducted. Comparisons and evaluations are notoriously hard for unsupervised problems like topic models. Different metrics and evaluations are discussed further under “Evaluation Metrics”. For this comparison, human evaluation was conducted. Meaning, that the generated topics was assessed and the topic model that produced the *best* topics was chosen for further use. *Best* means the most coherent and interpretable topics. The algorithms were tested using 10, 25 and 50 topics and the resulting topics were analyzed. Even though the topics generated by LSI may be mathematically correct they do not provide as much word coherence and interpretable topics. The result of this test agrees with [12] in that LDA generates more coherent and human interpretable topics. As mentioned, this was a quick analyzing test and the results should not be used to prove that LDA provides more coherent/interpretable topics than LSI. More analysis and parameter tweaking must be performed and the use of topic model depend on the needs of the user. The comparison between LSI and LDA is not the goal of this project, but a quick test had to be conducted to decide on the topic model to be used.

1) LDA

Latent Dirichlet Allocation *is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [10]*. This means that LDA assumes a document (a summary in this case) consist of several topics, where each topic consists of words with different probabilities of it belonging to that topic. As an example, water, fish, boat might be words with a high probability of belonging to a topic regarding fishing, or oceans. These words might occur together very frequently. LDA tries to find these latent topics. It is important to note that LDA does not provide names for topics but rather a topic is just a group of words. There exist techniques to automatically label topics, briefly discussed under "Conclusion". LDA is a fairly advanced topic model, and trying to explain it in detail is out of scope for this project. More information on LDA can be found in [10]. The most important parameter setting in topic modeling is the number of *topics.* This could be considered the same as the number of clusters in clustering algorithms. Finding the optimal number of topics is discussed more under "Evaluation Metrics".

2) LDA Implementation

LDA will be implemented with the help of a topic modeling library for python called Gensim [13]. Gensim provides, besides topic modeling implementations, also other useful functions like bag-of-words creation from documents as well as coherence evaluations. There are some needed parameters and configurations, some specific to Gensim, that needs to be discussed. Gensim provides two different modes of operation for LDA, called *batch* and *online* mode. In batch mode, Gensim will iterate through the entire training corpus (document collection) when training. For larger corpora, this requires a lot of resources (specifically memory and time). Online mode trains the model incrementally on small batches, which requires a lot less resources as well as enables the model to be updated by streaming data (which is not possible with batch mode without iterating through the entire dataset again). Online mode has been proven to find topics just as good, or better then batch mode, in a fraction of the time [14]. Thus, the online version of LDA will be used in this project.

One important parameter (belonging to Gensim and not the LDA algorithm) is *passes*. This parameter specifies the number of times LDA can see the corpus. This is especially useful for smaller corpora where one pass over the data might not be enough. As will be shown under "Results and Analysis" many passes are needed for this project since the number of tokens are drastically reduced after the pre-processing stage.

Another parameter is *iterations*, which defines the number of times the E-step will be executed for each document, this can be seen in algorithm 2 in [14]. There also exist two learning parameters called *alpha* and *eta* which are used influence the sparsity of the topic/word assignments. Alpha has been proven to be able to be learned from the data [15] and Gensim has an auto setting which will therefore be used. Eta is used for boosting the weights of some specified words in some topics, this is not necessary in this project and is therefore not used.

D. Evaluation Metrics

Evaluating topic models is difficult, since it depends on the type of problem that is the be solved. No universal evaluation metric exists. The most common way of evaluating is to measure the log-likelihood of a held-out test set. It has been shown that models who perform well on this metric, might generate less semantically meaningful topics [16].

Measuring coherence between words in topics is another measuring technique used to try to differentiate between good and bad topics where the coherence between words are quantified and measured. Different metrics for measuring coherence exists. According to [17] different coherence metrics perform differently depending on the data. Overall the CV metric seems to correlate best with human interpretation and is therefore the metric used in this project.

The gold standard in topic quality and interpretability evaluation is having humans evaluating. One approach to this is done by [16], where they introduce a couple of tests, *word intrusion* and *topic intrusion*. In the word intrusion test one intrusion word is added and the model is then evaluated based on whether the testers can find this word. Topic intrusion is the same approach but to introduce an intruder topic into a document and see whether it can be found. However, having humans evaluate each model would take an enormous amount of time and effort. Because of that, coherence tests will be done, alongside some human interpretation and evaluation (by the author). Coherence testing using the CV metric will be conducted to find the optimum number of topics, passes and iterations. The top resulting models will then be evaluated by the author. The results of these measurements can be found under "Results & Analysis".

E. Testing

Testing of the model will be done using a few known movie summaries. The results depend on how well the topical structure is inferred.

IV. RESULTS & ANALYSIS

The starting dataset containing 42,306 movie summaries was run through the specified pre-processing stages. The summaries contained around 74,4 million words initially (mostly duplicates and stopwords). After stopword removal, lemmatization, and filtering of nouns, 51,909 unique tokens remained. Filtering these tokens by removing words appearing in more than 60% and in less than 5 documents resulted in 13,749 unique tokens. These were mostly high quality common nouns in their lemmatized form.

The LDA model requires a corpus of bag-of-words, meaning that each document contains a “bag” of words, with no special ordering. These words are then marked by how frequently they appear in a document.

To find the optimal number of topics, a coherence score for each model had to be calculated. Luckily, gensim provides this feature. A grid search was done measuring the coherence score of each model, using topic numbers ranging from 20 to 70. The model was set to run at 15 passes and 50 iterations (default in gensim). The resulting graph can be seen in figure 1. The decision to use 20 topics as a lower bound comes from the fact that less than 20 topics would provide too general topics. 70 as an upper limit was decided because the resulting topics would result in topics being to “specialized” towards certain movies, and not general enough. The coherence scores pointed at models with 34 and 56 topics provided the most coherent topics, where 56 had slightly higher scores. The two models were then evaluated by the author and 56 was selected as the one providing higher coherence and interpretability. Note that since LDA is a probability model the topics may change slightly between runs, therefore coherence scores may differ.

A similar coherence test was done when finding the optimal number of passes. The model was set to use 56 topics and 50 iterations. The number of passes was ranging from 10-100 in steps of 5. Looking at the results, one can see that the coherence scores increases quite much at around 20 and then fluctuate. One conclusion that can be drawn from this is that over 25 passes are needed. The difference in coherence between 25 and 100 is not that big, however the computational time difference between 25 and 100 passes is enormous. Therefore, the number of passes to be used was decided to be 25. The resulting graph can be seen in figure 2.

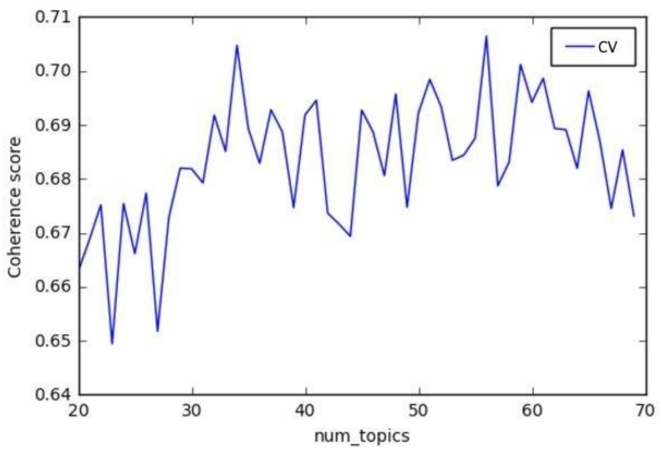

Figure 1. Coherence score of models set to 20 – 70 topics. Higher score means better results.

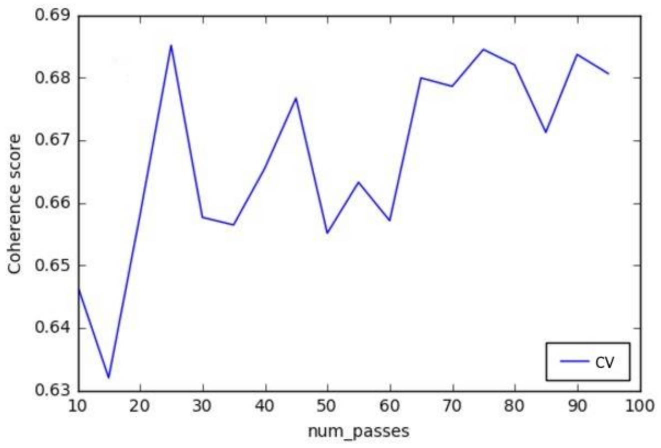

Figure 2. Coherence score for the optimal number of passes ranging from 10 to 100 in steps of 5. Higher score means better results.

The same procedure was conducted to find the optimal number of *iterations*. When running initial tests, iterations did not seem to make that much of a difference, as long as it was set higher than the gensim default of 50. The testing model was set to use 56 topics and 25 passes. Iterations were tested between 10 and 160 in steps of 50. As can be seen in figure 3, iterations peeks at around 60 and then declines.

The results of these coherence scores should be taken with a grain of salt. Even though they have been tested to correlate with human interpretability it is still not as good as having humans interpret the results. However, using coherence scores can give an approximation on where to start looking.

The final model was configured to have 56 topics, 25 passes and 60 iterations. This model does not have to be the optimal model. LDA yields different topic distributions each time. This means that running the tests again might yield slightly different results. Therefore, the model should be taken as an approximation. It is very hard, if not impossible, to find an optimal model considering humans interpret topics and their word distributions differently.

Running the model results in some very coherent topics, unfortunately some very hard to understand topics as well. One topic could easily be interpreted as “Studying and School” as its top ten words are: “school", "student", "teacher", class", “book", “college", "professor", "classmate", “university" and "film". Another one could be interpreted as “Crime / Murder” including words as “murder”, “case”, “killer”, “police”, “death”, “evidence”, “crime”, “victim”, “investigation” and “court”. Unfortunately, there are also some hard to interpret topics, one example is a topic including: “master”, “shark”, “art”, “skill” and more. There also exist topics which have mostly coherent words but includes some intruders: “town”, “bank”, “sheriff”, “robber”, “desert”, “robbery”, “townspeople”, “spider”, “deputy”, and “gun”. The topics can be seen in appendix A.

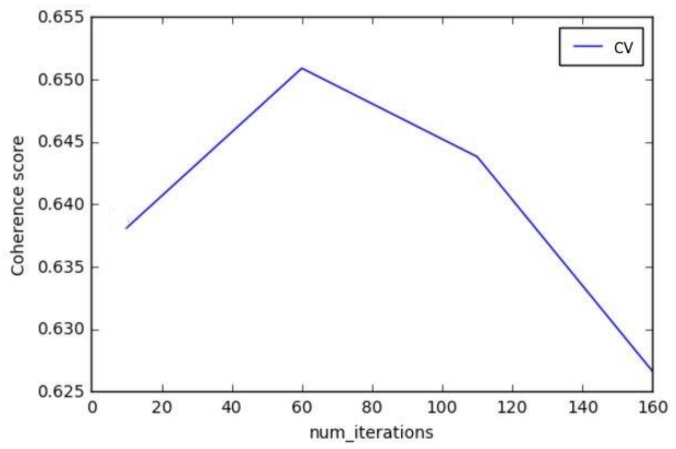

Figure 3. Coherence score for optimal number of iterations ranging from 10 to 160 in steps of 50. The higher value the better.

There could be several reasons for this. Not having enough or too highly scoped training data, which results in less coherent topics. The model could also need further tweaking. Inferring topics on an unseen movie highly depends on the quality of the its summary. Quality in this case, means the number of common nouns left in the document after pre-processing. This means that, even though some movie summaries are short, LDA can still provide decent topics if they contain more descriptive words.

Inferring topics on “Avatar” (an action/adventure/fantasy movie from 2009) results in several different topics. The highest probability topic includes words like: “soldier”, “battle”, “army”, “war” which is easily interpreted as a topic concerning war and battles. This holds true in the case of “Avatar” which involves a lot of battles. Another topic includes words as: “alien”, “space”, “planet” and can be interpreted as a topic regarding aliens and outer space. This also holds true to “Avatar”. Other topics could be interpreted as concerning relationships, group/tribes, and crime. Overall, the topic inference for “Avatar” is rather good. The results can be seen in appendix B.

Another movie tested was “Moneyball” (a biography/drama/sport movie from 2011). For this movie, the model inferred several topics but the highest probability topic had a lot higher probability than the second highest topic. The first having words like: “game”, “team”, “player”, “football”, “competition” and “coach” which is easily interpreted as “Sports”. Moneyball highly concerns sports. The other lower probability topics where topics regarding secret agents, affairs and drug money which might not fit the movie that well. Results can be seen in appendix C.

“Inside Job” is documentary regarding the financial crisis in 2008. The model infers topics regarding “City/Government” and “News” as the top probability topics, which indeed reflects some of the content in the documentary. It also infers a hard to interpret topic including words like: “team”, “agent”, “spy” and “formula”. A topic regarding finance or money would be expected here but was not inferred. The reason for this, looking through the generated topics, is that no topic exists regarding these two areas (except drug money). Some reasons for this might be the lack of training data concerning these areas, the need for more topics, or that “money” is too general and exist in many different topics. The results for “Inside Job” can be seen in appendix D.

One reason the model infers topics not belonging, could be the testing data. If the summary is too short and contains too few descriptive words, the model is having a hard time inferring correct topics since it is not given enough information. As mentioned previously, the topic quality and coherence highly affects the result since these topics are the ones that will be inferred on the testing data.

V. CONCLUSION

The goal of this project was to find a model which could infer topics on movies. The resulting model succeeds in doing so, however it is not perfect. Even though it manages to find topics which represents parts of the story in the testing movies, it also infers topics not true to those movie, albeit usually with lower probability. As mentioned, there could be several reasons for this with the most likely being not enough or too highly scoped training data.

One important lesson learnt during this project is “garbage-in and garbage-out”. I learned this the hard way during the project, with many different tests in pre-processing the data. A lot of time was spent on this before proceeding towards model decision and evaluation.

Topic modelling is an amazing technique for describing and inferring topics on data. The result of this project might not be fit for industrial use, as is. However, it could be used as a base for implementing recommendation systems or be used alongside a supervised system to automatically label the found topics. One approach in automatically labeling topics was done by [18] using a combination of association measures and lexical features fed into a supervised ranking model. This kind of method could also counteract intruder words in topics.

VI. IMPROVEMENTS & FUTURE WORK

Some suggestions exist that may be able to improve the model. Considering *n-grams* could result in more descriptive and interpretable topics, since this enables words commonly used together to also appear together [19]. Meaning words like “World War 2” could be generated by the model, instead of “war”, “2” and “world”. However, this also increases the computing time by a considerable amount. There is also the concept of *chunking,* where the summaries would be split into pieces and processed as individual documents. This could improve the separation of topics. For this project, the summaries were of many different lengths and finding an accurate length of the chunks is difficult.

REFERENCES

[1] "Topic model", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Topic_model. [Accessed: 30- Dec- 2016].

[2] S. Bergamaschi and L. Po, "Comparing LDA and LSA Topic Models for Content-Based Movie Recommendation Systems", Lecture Notes in

Business Information Processing, pp. 247-263, 2015. Doi: 10.1007/978-3-319-27030-2_16

[3] A. Bhowmick, U. Prasad and S. Kottur, “Movie Recommendation based on Collaborative Topic Modeling”, 2016 [Online] Available: https://satwikkottur.github.io/reports/F14-ML-Report.pdf [Accessed: 30-Dec-2016].

[4] D. Bamman, B. O'Connor, and N. Smith, “Learning Latent Personas of Film Characters”, ACL 2013, Sofia, Bulgaria, August 2013.

[5] P. Lison and Jörg Tiedemann, “OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles”. In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016).

[6] “Expanded Stopwords List Matthew L. Jockers", Matthewjockers.net, 2016. [Online]. Available: http://www.matthewjockers.net/macroanalysisbook/expanded-stopwords-list/. [Accessed: 30- Dec- 2016].

[7] "Part-of-speech tagging", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Part-of-speech_tagging. [Accessed: 30-Dec- 2016].

[8] "nltk.tag.perceptron — NLTK 3.0 documentation", Nltk.org, 2016. [Online]. Available: http://www.nltk.org/_modules/nltk/tag/perceptron.html. [Accessed: 30-Dec- 2016].

[9] "nltk.stem.wordnet — NLTK 3.0 documentation", Nltk.org, 2016. [Online]. Available: http://www.nltk.org/_modules/nltk/stem/wordnet.html. [Accessed: 30-Dec- 2016].

[10] D. Blei, A. Ng and M. Jordan, "Latent dirichlet allocation", The Journal of Machine Learning Research, vol. 3, 312003, pp. 993-1022, 2003.

[11] T. Landauer, P. Foltz and D. Laham, "An introduction to latent semantic analysis", Discourse Processes, vol. 25, no. 2-3, pp. 259-284, 1998.

[12] K. Stevens, P. Kegelmeyer, D. Andrzejewski and D. Buttler, "Exploring topic coherence over many models and many topics", EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 952-961, 2012.

[13] "gensim: topic modelling for humans", Radimrehurek.com, 2016. [Online]. Available: https://radimrehurek.com/gensim/. [Accessed: 30-Dec- 2016].

[14] D. Blei, M. Hoffman and F. Bach, "Online Learning for Latent Dirichlet Allocation", NIPS'10 Proceedings of the 23rd International Conference on Neural Information Processing Systems, pp. 856-864, 2010.

[15] H. Wallach, D. Mimno and A. McCallum, "Rethinking LDA: Why Priors Matter", NIPS'09 Proceedings of the 22nd International Conference on Neural Information Processing Systems, pp. 1973-1981, 2009.

[16] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. Blei. “Reading Tea Leaves: How Humans Interpret Topic Models”. Neural Information Processing Systems, 9 pages, 2009.

[17] M. Röder, A. Both and A. Hinneburg, "Exploring the Space of Topic Coherence Measures", WSDM '15 Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, pp. 399-408, 2015.

[18] J. Lau, K. Greiser, D. Newman and T. Baldwin, "Automatic Labelling of Topic Models", HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, pp. 1536-1545, 2011.

[19] "N-gram", En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/N-gram. [Accessed: 30- Dec- 2016].

Appendix

A. Results of running the Topic model

```
[(0,
 '0.041*"doctor" + 0.040*"hospital" + 0.027*"life" + 0.024*"relationship" + '
 '0.023*"daughter" + 0.019*"time" + 0.016*"marriage" + 0.016*"parent" + '
 '0.016*"patient" + 0.014*"year"'),
(1,
 '0.127*"train" + 0.067*"station" + 0.047*"bus" + 0.041*"driver" + '
 '0.024*"car" + 0.019*"passenger" + 0.015*"taxi" + 0.012*"track" + '
 '0.011*"railway" + 0.011*"board"'),
(2,
 '0.037*"letter" + 0.037*"job" + 0.027*"relationship" + 0.024*"life" + '
 '0.022*"company" + 0.021*"friend" + 0.019*"store" + 0.017*"boyfriend" + '
 '0.017*"time" + 0.015*"woman"'),
(3,
 '0.064*"time" + 0.023*"bank" + 0.020*"diamond" + 0.018*"memory" + '
 '0.012*"future" + 0.011*"vault" + 0.009*"travel" + 0.009*"security" + '
 '0.008*"order" + 0.007*"number"'),
(4,
 '0.034*"body" + 0.029*"room" + 0.018*"death" + 0.014*"head" + 0.012*"face" + '
 '0.012*"hand" + 0.011*"hospital" + 0.010*"knife" + 0.008*"demon" + '
 '0.008*"doll"'),
(5,
 '0.107*"gold" + 0.065*"treasure" + 0.044*"map" + 0.026*"coin" + 0.018*"duck" '
 '+ 0.015*"chip" + 0.013*"adventure" + 0.011*"location" + 0.010*"duo" + '
 '0.010*"piece"'),
(6,
 '0.345*"child" + 0.045*"web" + 0.024*"story" + 0.016*"cult" + 0.013*"statue" '
 '+ 0.013*"kid" + 0.013*"childrens" + 0.012*"film" + 0.011*"adult" + '
 '0.008*"couple"'),
(7,
 '0.135*"ghost" + 0.047*"baseball" + 0.024*"bat" + 0.021*"ice" + 0.016*"vote" '
 '+ 0.016*"cream" + 0.013*"game" + 0.012*"angel" + 0.010*"plate" + '
 '0.009*"skull"'),
(8,
 '0.139*"prison" + 0.052*"guard" + 0.048*"prisoner" + 0.027*"jail" + '
 '0.026*"cell" + 0.025*"inmate" + 0.023*"escape" + 0.019*"sentence" + '
 '0.017*"camp" + 0.015*"release"'),
(9,
 '0.168*"girl" + 0.073*"boy" + 0.063*"group" + 0.033*"camp" + 0.024*"sister" '
 '+ 0.020*"friend" + 0.010*"party" + 0.008*"counselor" + 0.008*"plan" + '
 '0.007*"camper"'),
(10,
 '0.045*"power" + 0.035*"alien" + 0.032*"planet" + 0.029*"space" + '
 '0.028*"scientist" + 0.027*"machine" + 0.024*"robot" + 0.020*"human" + '
 '0.019*"earth" + 0.017*"experiment"'),
(11,
 '0.075*"spirit" + 0.043*"soul" + 0.039*"death" + 0.030*"mountain" + '
 '0.028*"body" + 0.021*"asylum" + 0.015*"year" + 0.014*"god" + 0.014*"curse" '
 '+ 0.011*"people"'),
(12,
 '0.030*"water" + 0.025*"group" + 0.018*"zombie" + 0.014*"lake" + '
 '0.013*"cave" + 0.012*"survivor" + 0.011*"river" + 0.011*"boat" + '
 '0.009*"egg" + 0.008*"attack"'),
(13,
 '0.078*"band" + 0.077*"music" + 0.048*"singer" + 0.028*"concert" + '
 '0.026*"club" + 0.021*"performance" + 0.021*"rock" + 0.020*"stage" + '
 '0.017*"group" + 0.017*"record"'),
(14,
 '0.136*"film" + 0.039*"story" + 0.031*"character" + 0.025*"scene" + '
 '0.024*"book" + 0.021*"movie" + 0.012*"life" + 0.010*"event" + '
 '0.009*"people" + 0.009*"series"'),
(15,
 '0.183*"town" + 0.037*"bank" + 0.028*"sheriff" + 0.020*"robber" + '
 '0.018*"desert" + 0.014*"robbery" + 0.012*"townspeople" + 0.011*"spider" + '
 '0.010*"deputy" + 0.009*"gun"'),
(16,
 '0.149*"baby" + 0.061*"church" + 0.049*"farm" + 0.030*"painting" + '
 '0.029*"tree" + 0.015*"farmer" + 0.014*"child" + 0.014*"cottage" + '
 '0.013*"birth" + 0.012*"mirror"'),
(17,
 '0.054*"vampire" + 0.048*"blood" + 0.039*"body" + 0.023*"woman" + '
 '0.021*"box" + 0.018*"coffin" + 0.014*"museum" + 0.014*"kidnapper" + '
 '0.013*"victim" + 0.011*"corpse"'),
(18,
 '0.045*"hotel" + 0.032*"wife" + 0.020*"job" + 0.017*"affair" + '
 '0.015*"husband" + 0.012*"room" + 0.011*"divorce" + 0.009*"contract" + '
 '0.009*"business" + 0.009*"secretary"'),
(19,
 '0.107*"truck" + 0.033*"park" + 0.032*"circus" + 0.030*"ball" + 0.026*"ring" '
 '+ 0.016*"trailer" + 0.014*"ride" + 0.013*"road" + 0.013*"monkey" + '
 '0.012*"clown"'),
(20,
 '0.026*"castle" + 0.019*"power" + 0.016*"palace" + 0.015*"order" + '
 '0.014*"sword" + 0.012*"life" + 0.011*"kingdom" + 0.010*"spell" + '
 '0.010*"battle" + 0.009*"time"'),
(21,
 '0.041*"door" + 0.032*"car" + 0.032*"room" + 0.020*"window" + 0.017*"gun" + '
 '0.013*"floor" + 0.012*"fire" + 0.010*"wall" + 0.010*"head" + 0.008*"key"'),
(22,
 '0.060*"master" + 0.030*"shark" + 0.027*"art" + 0.021*"skill" + '
 '0.021*"fight" + 0.017*"clan" + 0.014*"monk" + 0.013*"training" + '
 '0.012*"mine" + 0.012*"sword"'),
(23,
 '0.054*"grandfather" + 0.031*"orphan" + 0.030*"orphanage" + 0.029*"snake" + '
 '0.019*"woman" + 0.018*"nun" + 0.016*"story" + 0.015*"child" + '
 '0.012*"convent" + 0.011*"people"'),
(24,
 '0.120*"village" + 0.048*"daughter" + 0.037*"brother" + 0.028*"villager" + '
 '0.017*"marriage" + 0.015*"money" + 0.015*"story" + 0.014*"people" + '
 '0.013*"family" + 0.010*"twin"'),
(25,
 '0.069*"grandmother" + 0.059*"bear" + 0.050*"wolf" + 0.047*"witch" + '
 '0.034*"fashion" + 0.028*"piano" + 0.023*"model" + 0.019*"potion" + '
 '0.014*"designer" + 0.011*"deer"'),
(26,
 '0.036*"men" + 0.012*"ranch" + 0.012*"bandit" + 0.011*"land" + 0.010*"gun" + '
 '0.009*"wife" + 0.009*"town" + 0.009*"death" + 0.008*"order" + '
 '0.008*"border"'),
(27,
 '0.076*"radio" + 0.014*"guitar" + 0.012*"time" + 0.011*"station" + '
 '0.010*"broadcast" + 0.009*"music" + 0.008*"cigarette" + 0.008*"program" + '
 '0.008*"life" + 0.007*"milk"'),
(28,
 '0.119*"life" + 0.022*"city" + 0.022*"story" + 0.020*"family" + '
 '0.017*"people" + 0.017*"film" + 0.015*"year" + 0.015*"journey" + '
 '0.013*"friend" + 0.010*"childhood"'),
(29,
 '0.040*"parent" + 0.038*"friend" + 0.025*"time" + 0.022*"kid" + '
 '0.020*"family" + 0.019*"car" + 0.018*"party" + 0.013*"school" + '
 '0.010*"trip" + 0.008*"life"'),
(30,
 '0.066*"plane" + 0.038*"pilot" + 0.030*"flight" + 0.021*"aircraft" + '
 '0.019*"crash" + 0.019*"mission" + 0.014*"air" + 0.014*"base" + '
 '0.013*"rocket" + 0.012*"missile"'),
(31,
 '0.108*"police" + 0.022*"officer" + 0.021*"crime" + 0.021*"cop" + '
 '0.017*"gangster" + 0.015*"thief" + 0.014*"criminal" + 0.014*"wife" + '
 '0.012*"bos" + 0.012*"gun"'),
(32,
 '0.076*"team" + 0.048*"agent" + 0.026*"spy" + 0.014*"formula" + '
 '0.011*"order" + 0.009*"list" + 0.008*"plan" + 0.007*"president" + '
 '0.007*"intelligence" + 0.007*"office"'),
(33,
 '0.066*"murder" + 0.042*"case" + 0.038*"killer" + 0.033*"police" + '
 '0.022*"death" + 0.019*"evidence" + 0.019*"crime" + 0.018*"victim" + '
 '0.017*"investigation" + 0.017*"court"'),
(34,
 '0.068*"woman" + 0.034*"sex" + 0.032*"apartment" + 0.027*"phone" + '
 '0.020*"room" + 0.018*"party" + 0.017*"call" + 0.015*"bar" + 0.014*"couple" '
 '+ 0.011*"men"'),
(35,
 '0.094*"game" + 0.069*"team" + 0.038*"player" + 0.026*"football" + '
 '0.026*"competition" + 0.021*"coach" + 0.012*"match" + 0.011*"basketball" + '
 '0.011*"time" + 0.008*"sport"'),
(36,
 '0.149*"race" + 0.131*"horse" + 0.014*"gypsy" + 0.012*"track" + '
 '0.012*"lottery" + 0.012*"barber" + 0.011*"rider" + 0.011*"prize" + '
 '0.008*"racer" + 0.008*"racing"'),
(37,
 '0.049*"beach" + 0.036*"oil" + 0.035*"plant" + 0.026*"water" + '
 '0.023*"worker" + 0.019*"construction" + 0.014*"sea" + 0.011*"company" + '
 '0.010*"site" + 0.010*"resort"'),
(38,
 '0.047*"government" + 0.024*"city" + 0.022*"computer" + 0.016*"virus" + '
 '0.014*"country" + 0.014*"people" + 0.014*"official" + 0.014*"resistance" + '
 '0.013*"group" + 0.009*"war"'),
(39,
 '0.057*"soldier" + 0.036*"battle" + 0.035*"army" + 0.028*"war" + '
 '0.023*"attack" + 0.023*"force" + 0.019*"enemy" + 0.017*"group" + '
 '0.016*"troop" + 0.014*"order"'),
(40,
 '0.061*"apartment" + 0.026*"dream" + 0.022*"building" + 0.021*"client" + '
 '0.016*"work" + 0.014*"woman" + 0.012*"prostitute" + 0.012*"job" + '
 '0.009*"business" + 0.009*"street"'),
(41,
 '0.153*"ship" + 0.083*"crew" + 0.027*"captain" + 0.022*"pirate" + '
 '0.018*"board" + 0.016*"submarine" + 0.013*"boat" + 0.012*"sea" + '
 '0.012*"sailor" + 0.010*"deck"'),
(42,
 '0.120*"money" + 0.054*"car" + 0.013*"drug" + 0.013*"job" + 0.012*"debt" + '
 '0.010*"cash" + 0.007*"bank" + 0.007*"dollar" + 0.007*"casino" + '
 '0.006*"order"'),
(43,
 '0.046*"cat" + 0.026*"mouse" + 0.020*"head" + 0.016*"hole" + 0.012*"cartoon" '
 '+ 0.012*"bird" + 0.011*"time" + 0.011*"tree" + 0.010*"tail" + '
 '0.008*"ground"'),
(44,
 '0.237*"gang" + 0.052*"member" + 0.039*"leader" + 0.018*"thug" + '
 '0.016*"street" + 0.015*"motorcycle" + 0.014*"group" + 0.013*"fight" + '
 '0.012*"plan" + 0.011*"tiger"'),
(45,
 '0.086*"fight" + 0.055*"match" + 0.024*"campaign" + 0.018*"opponent" + '
 '0.017*"challenge" + 0.017*"fighter" + 0.016*"champion" + 0.016*"manager" + '
 '0.016*"boxer" + 0.016*"round"'),
(46,
 '0.058*"film" + 0.030*"movie" + 0.027*"role" + 0.026*"actor" + '
 '0.021*"director" + 0.020*"play" + 0.017*"stage" + 0.016*"dance" + '
 '0.016*"actress" + 0.014*"studio"'),
(47,
 '0.044*"story" + 0.038*"reporter" + 0.036*"newspaper" + 0.023*"villain" + '
 '0.021*"minister" + 0.021*"journalist" + 0.020*"hero" + 0.019*"news" + '
 '0.017*"politician" + 0.015*"people"'),
(48,
 '0.129*"dog" + 0.070*"monster" + 0.068*"creature" + 0.048*"animal" + '
 '0.019*"lion" + 0.018*"werewolf" + 0.013*"elephant" + 0.013*"beast" + '
 '0.012*"lab" + 0.011*"balloon"'),
(49,
 '0.157*"family" + 0.074*"wife" + 0.060*"husband" + 0.040*"daughter" + '
 '0.022*"life" + 0.020*"woman" + 0.018*"marriage" + 0.015*"child" + '
 '0.013*"year" + 0.012*"son"'),
(50,
 '0.163*"school" + 0.101*"student" + 0.047*"teacher" + 0.046*"college" + '
 '0.036*"class" + 0.017*"professor" + 0.015*"friend" + 0.012*"classmate" + '
 '0.010*"university" + 0.009*"film"'),
(51,
 '0.022*"drug" + 0.021*"bomb" + 0.017*"agent" + 0.014*"gun" + '
 '0.011*"terrorist" + 0.011*"police" + 0.010*"hostage" + 0.010*"officer" + '
 '0.010*"group" + 0.010*"assassin"'),
(52,
 '0.116*"men" + 0.041*"war" + 0.037*"officer" + 0.019*"soldier" + '
 '0.016*"order" + 0.009*"duty" + 0.009*"veteran" + 0.009*"woman" + '
 '0.008*"slave" + 0.008*"fire"'),
(53,
 '0.067*"uncle" + 0.042*"factory" + 0.034*"aunt" + 0.033*"town" + '
 '0.025*"worker" + 0.020*"mayor" + 0.017*"community" + 0.010*"union" + '
 '0.010*"party" + 0.010*"time"'),
(54,
 '0.080*"wedding" + 0.031*"mansion" + 0.022*"estate" + 0.018*"marriage" + '
 '0.018*"guest" + 0.017*"ceremony" + 0.017*"necklace" + 0.016*"bride" + '
 '0.016*"party" + 0.015*"plan"'),
(55,
 '0.102*"island" + 0.053*"boat" + 0.028*"sea" + 0.026*"tribe" + '
 '0.015*"native" + 0.015*"jungle" + 0.015*"expedition" + 0.013*"people" + '
 '0.012*"land" + 0.010*"group"')]
```

B. ”Avatar” topics

```
Probability:  0.399025825445  :
 0.057*"soldier" + 0.036*"battle" + 0.035*"army" + 0.028*"war" + 0.023*"attack" + 0.023*"force"

Probability:  0.135196447951  :
 0.041*"doctor" + 0.040*"hospital" + 0.027*"life" + 0.024*"relationship" + 0.023*"daughter" + 0.019*"time"

Probability:  0.120930872742  :
 0.108*"police" + 0.022*"officer" + 0.021*"crime" + 0.021*"cop" + 0.017*"gangster" + 0.015*"thief"

Probability:  0.117615171632  :
 0.045*"power" + 0.035*"alien" + 0.032*"planet" + 0.029*"space" + 0.028*"scientist" + 0.027*"machine"

Probability:  0.0991440528871  :
 0.102*"island" + 0.053*"boat" + 0.028*"sea" + 0.026*"tribe" + 0.015*"native" + 0.015*"jungle"

Probability:  0.0553063745828  :
 0.075*"spirit" + 0.043*"soul" + 0.039*"death" + 0.030*"mountain" + 0.028*"body" + 0.021*"asylum"

Probability:  0.0445554483084  :
 0.045*"hotel" + 0.032*"wife" + 0.020*"job" + 0.017*"affair" + 0.015*"husband" + 0.012*"room"
```

C. “Moneyball” topics

```
Probability:  0.525380416669  :
 0.094*"game" + 0.069*"team" + 0.038*"player" + 0.026*"football" + 0.026*"competition" + 0.021*"coach"

Probability:  0.204764317729  :
 0.076*"team" + 0.048*"agent" + 0.026*"spy" + 0.014*"formula" + 0.011*"order" + 0.009*"list"

Probability:  0.183507928925  :
 0.135*"ghost" + 0.047*"baseball" + 0.024*"bat" + 0.021*"ice" + 0.016*"vote" + 0.016*"cream"

Probability:  0.0441986836767  :
 0.045*"hotel" + 0.032*"wife" + 0.020*"job" + 0.017*"affair" + 0.015*"husband" + 0.012*"room"

Probability:  0.0274597129092  :
 0.120*"money" + 0.054*"car" + 0.013*"drug" + 0.013*"job" + 0.012*"debt" + 0.010*"cash"
```

D. “Insider Job” topics

```
Probability:  0.294280585817  :
 0.047*"government" + 0.024*"city" + 0.022*"computer" + 0.016*"virus" + 0.014*"country" + 0.014*"people"

Probability:  0.187409339989  :
 0.044*"story" + 0.038*"reporter" + 0.036*"newspaper" + 0.023*"villain" + 0.021*"minister" + 0.021*"journalist"

Probability:  0.176865664067  :
 0.136*"film" + 0.039*"story" + 0.031*"character" + 0.025*"scene" + 0.024*"book" + 0.021*"movie"

Probability:  0.166857846509  :
 0.076*"team" + 0.048*"agent" + 0.026*"spy" + 0.014*"formula" + 0.011*"order" + 0.009*"list"

Probability:  0.0826733024264  :
 0.037*"letter" + 0.037*"job" + 0.027*"relationship" + 0.024*"life" + 0.022*"company" + 0.021*"friend"
```

```
Probability:  0.0513288456062  :
 0.163*"school" + 0.101*"student" + 0.047*"teacher" + 0.046*"college" + 0.036*"class" + 0.017*"professor"
```