TV-Shows Subtitles Analysis & Recommender System

Big Data Class Project 2014

Final Report

Claire Musso
Florian Simond
Grigory Rozhdestvenskiy
Khalil Hajji

Nassim Drissi El Kamili
Nils Bouchardon
Simon-Pierre Génot
Raphaël von Aarburg

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

1. Project Achievements

i. Crawling

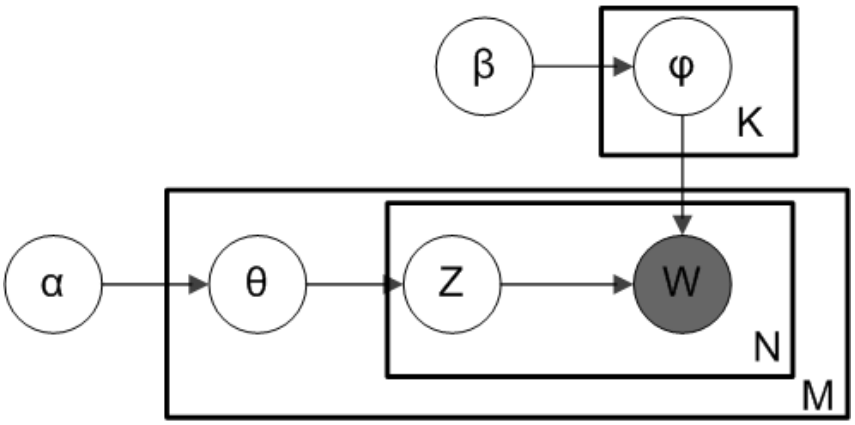
We first needed to select the adequate platform to crawl. We decided to use *tvsubtitles.net* since there was no limitations or whatsoever for our purposes. We used python scripting and successfully obtained the desired subtitles which we cleaned from duplicates by using once again python scripting.
We also wanted some other kind of metrics for our subtitle analysis like *IMDB-ratings*, *plots* of the TV shows, *posters* and other meaningful informations to be displayed. When they were available, we obtained them from *omdbapi.com.*
In total, we gathered subtitles and information on about 1121 shows.

ii. Preprocessing

We successfully parsed and reconstructed full sentences from subtitles files. We performed part-of-speech tagging, and selected only meaningful words for our analysis. That is, we kept: *adjectives*, *comparatives*, *superlatives*, singular and plural *nouns*, *proper nouns*, *verbs* in all their forms. We performed lemmatizing on these to reduce the size of the vocabulary: verbs were mapped to their infinitive form and noun to their singular. We also removed out of the dictionary words. Subtitles are handmade, and therefore contain a lot of mistakes, and misspelled words. Moreover, oral language is not as strict as the written language and therefore we also have a lot of non-existing words. We filtered the words using a dictionary, to keep only a good set of words. According to the creator of the LDA, removing words that appear in more than 90% of the shows helps the algorithm to give good results, so we also performed this step. Once that was done, we indexed all remaining words and counted their occurrences in a given show. In the end each show was represented by the vector of frequencies for each word in vocabulary.
In summary, after PoS filtering we had a raw vocabulary of around 550’000 words and reduced it to 54’000 words.

iii. Processing

In this part of the project our goal was to extract topics from the subtitles to be able to model our TV series. This problem, part of a subject called Topic Modelling, has gotten a lot of attention recently. The first paper by Deerwester [1] published in 1988 proposed the first solution to discovering topics in a corpus of documents : LSI. Later other solutions were proposed including pLSI [2]. The most recent technique and the most advanced is LDA, or Latent Dirichlet Allocation. Proposed by Blei and al. in 2003, this technique has been applied in many domains, including scientific topics, images or Twitter data.
The technique is based on a generative model that describes how documents inside the corpus are generated:



Source :http://en.wikipedia.org/wiki/File:Smoothed_LDA.png

In this model, K denotes the number of topics, N — the number of words inside a document, and M — the number of documents in the corpus. In this mixture model where the topic distribution and the per-topic word distribution are assumed to follow the Dirichlet prior.
In the previous image, alpha is the parameter of the Dirichlet prior on the per-document topic distributions, beta is the parameter of the Dirichlet prior on the per-topic word distribution (hyperparameters). Theta is the word topic distribution for document (one per document). Phi is the word distribution (one per topic). Z is the topic variable for each word, and W is the word itself (both z and w are sampled for each word).
Now that we have specified a model for generating documents, our job is to infer the parameters to learn the topics.
Blei's orginal paper used Variational Inference to do this, but other methods have been proposed, most notably Gibb's sampling, which is a MCMC optimization technique (Markov Chain Monte Carlo).
In past years, many advances have been made to scale this algorithm in a parallelized fashion, notably using Map-Reduce and Hadoop. This is a hard problem as there is a lot a shared information between different mappers in the algorithm. Nevertheless, using various optimization, some teams were able to propose efficient solutions. The first, lead by a team of Yahoo! labs, successfully parallelized Gibb's sampling [4]. More recently, a solution called Mr LDA was proposed for Variational Inference [5].
Variational Inference, though more complex than Gibb's sampling, permits a better level of precision by allowing the optimization of the alpha parameter. This is why we have chosen this version of LDA to implement in our case.
We designed our algorithm using Hadoop based on [5], and managed to recreate a running version of the algorithm. We put a lot of efforts in optimizing and improving the performance of our algorithm to be able to run multiple tests. At the first version of our code, we needed 35 minutes to run one job, and as we needed around 40 iterations for the algorithm to converge, the whole algorithm took us around 23 hours. At the end, we succeeded to decrease the running time by a factor of 10 to end up with 3 minutes by job which gives 2 hours for the whole algorithm.

iii. Recommender System

The recommender system was the last part of our project, its aim was to visualize the results of our LDA processing in an understandable way.
The user inputs a certain number of TV shows he wants a recommendation for, let's call this set T. For each show x in T, we compute the similarity of x with all the available TV shows in our database, sort those result (by similarity) and output k recommendations with maximum similarity. By doing so, we obtain for each show in T, a list of k recommendations.
We finally pick the k TV shows, among all those lists, that have the maximum similarity value. We reject all TV shows that were originally in X (ensuring that the user cannot be recommended a TV show for which he asked a recommendation).
The similarity between TV-shows $X, Y \in T$ (represented in this case in their vector form, namely their topic distribution) is computed as:

$$sim(X, Y) = \frac{X \cdot Y}{||X|| \cdot ||Y||}$$

If requested by the user, the result is then sorted by IMDB rating.

2. Results

i. Raw

The raw results can be found in the repository of the projects under the processing/results/ subfolders.
In here, you will find different files :

- vocabulary: the mapping of all different words from subtitles to numeric ids
- shows: the mapping of all shows to numeric ids
- lambda: topics distribution over vocabulary (each column is one topic)
- gamma: topic proportions for each show (one show per line)
- alpha, gradient: LDA parameter files (can be disregarded)

ii. LDA results :

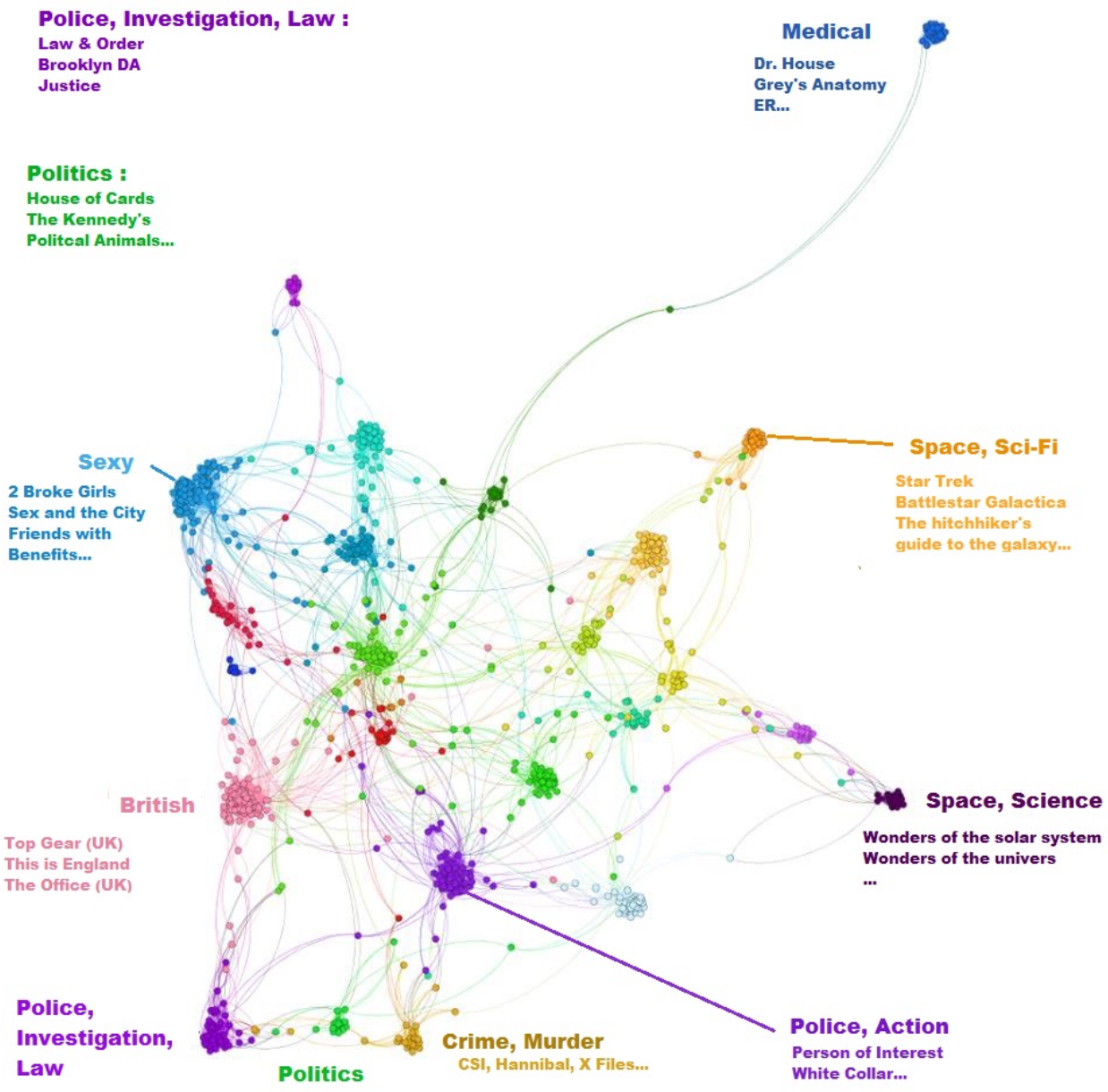
We have obtained results we consider to be very good. After testing the results for a number of different choices for the number of topics, we settled on 25. Below we have included some of the topics we have obtained, based on names we have given them by analysing their bag of word representations :

1 Vice
2 College Life
3 Vampire / Supernatural
6 Music
7 Power
8 Cartoon / (Family)
9 Space / Science
10 Dance / Contest
11 Dynasty
12 Space War / Science Fiction
13 Magic
14 Police case / Investigation / Law
15 British
16 Army
17 Family
18 Medical
19 Noisy
20 Sexy
21 Relationship
22 Comics / Technology
23 Politics
24 Crime / Murder
25 Police / Action

It is interesting to notice that most topics seem obvious and could correspond to the IMDB classification (Medical, Army, Relationship…). Others are more subtle. For example, our algorithm recognized a whole class of shows that had a particularity in their scripts : TV Shows from the U.K. ! It should be noticed however that topic naming is just done for demonstrational purposes. The topics discovered by LDA in general don't represent some particular theme that can be easily named. The topic is the distribution over all 55,000 words in the vocabulary, so it's impossible to convey all information that this distribution represents in one name.
In our example, the algorithm divided one of the most common topic in TV Shows: police and crime. There are 3 topics that are related to this theme, but they differ slightly: it looks like one describes crime and murder, legal investigations, or just police action. These results are very encouraging.
Below we included the top 10 words from a topic that we've called "Police/Action". And the shows that have the biggest proportion of this topic. We named the topic just by looking at the top words, and later found that these shows can be really described well as Police/Action.

Words :	Shows :
car	The Bridge
cop	The Shield
gun	Dark Blue
detective	Day Break
police	The Chicago Code
shoot	The Good Guys
phone	Southland
brother	Adam-12
streat	K-Ville
steal	NYC 22

In order to visualize what all this information means, we have created the 5-nearest neighbors graph of the results, that we show below :



This graph, very useful in our case, was constructed in the following way : each node is linked to it's 5 closest neighbors in terms of cosine similarity. This means that for every node, every one of it's neighbor will be in the output of our recommender system. Because there are so many shows and links, we have spatialized our graph in such a clustered fashion using a force field visualisation technique (using Gephi). Each color represents the top topic of the node.

The first observation on this graph is that the nodes are very well clustered into topics. For example, the Medical topic only has two links that point outside of it.

The second observation is that for the topics that are all related to Police, Crime, etc, they are very close together : indeed the algorithm has put the shows in separate clusters, but still gives many links going from one to the other. This is good because it means that our method is sufficiently flexible.
A surprising observation though comes from the complete separation of the Space & Science topic with the Space & Sci-Fi topic. We could have forseen them to be close together, but it seems this separation does make sense as the shows in each category have very little in common.

ii. Website

We choose to implement a GUI for our recommender under a website form. We implemented the back end with the Play! Framework which is coded in Java while the front end is made of with HTML and CSS.
It works as follow. It opens on an index page on which you can select the shows you want the recommendations for. Once it is done, you click the recommend button and a list of most similar shows to the given ones appears. One can then refine those results by ordering them in function of their IMDB ratings, can view a show presentation page by clicking the name of the latter or do the same with a topic.

iii. Visuals

We tried to get the most from our results in order to present them in a fancy way. To do so, we have created different visuals.
First, we created a pie chart displayed on every show page which highlight the proportion of topics in the show. We also created a cloud of words for every show/topic. The latter presents a bag of the most important words of a given show/topic by printing them in function of their importance in the show/topic. Note that for a show, the colours used in the pie charts are re used in the cloud of words in order to identify which topic a word belongs to.

3. Examples of Results

Below, we give some examples of recommendations :

- Game of Thrones :

Title	Similarity	IMDB Rating
Crusoe	90.4 %	6.9
Krod Mandoon and the Flaming Sword of Fire	90.36 %	NA
1066 The Battle for Middle Earth	89.49 %	NA
Divine? The Series	89.01 %	NA
Kung Fu	87.93 %	7.8
Roar	86.81 %	7.5
Neverwhere	85.84 %	7.3
The Pillars Of The Earth	84.46 %	8.2
Rome	83.69 %	9.0
Poltergeist The Legacy	83.54 %	NA
Atlantis	83.27 %	6.5
Reign	81.94 %	7.9
Ancient Rome - The Rise and Fall of an Empire	81.49 %	NA
Kings	80.69 %	8.3
Thor & Loki Blood Brothers	80.58 %	NA

- Dexter :

Title	Similarity	IMDB Rating
The Glades	86.32 %	7.5
Rizzoli and Isles	83.12%	NA
CSI Miami	82.5%	NA
Killer Insctinct	82.45%	7.4
Crossing Jordan	81.92%	6.9
The Forgotten	80.45%	5.8
CSI NY	78.99%	NA
Saving Grace	78.3%	6.9
Criminal Minds	78.07%	8.2
Profiler	77.73%	7.3
CSI	77.64%	7.9
Castle	77.34%	8.4
Body Of Proof	75.37%	7.0
Hawaii Five-0	75.29 %	7.5
Criminal Minds? Suspect Behavior	73.94 %	NA

- The Simpsons :

Title	Similarity	IMDB Rating
Men Behaving Badly	99.39 %	6.7
Beavis and Butt-Head	99.22 %	7.6
The Cleveland Show	99.03 %	5.6
Family Guy	98.04 %	8.5
The Penguins Of Madagascar	97.59 %	7.7
Robot Chicken	97.45 %	7.8
American Dad!	97.32 %	7.7
Futurama	97.28 %	8.7
My Name Is Earl	95.2 %	7.9
South Park	95.2 %	8.9
Raising Hope	94.98 %	8.0
Neighbors From Hell	94.75 %	6.0
The Ren & Stimpy Show	93.94 %	7.6
Clerks	93.63 %	7.4
Key And Peele	93.46 %	8.2

4. References

1. Deerwester, S., et al, Improving Information Retrieval with Latent Semantic Indexing, Proceedings of the 51st Annual Meeting of the American Society for Information Science 25, 1988, pp. 36–40.
2. Thomas Hofmann, *Learning the Similarity of Documents : an information-geometric approach to document retrieval and categorization*, Advances in Neural Information Processing Systems 12, pp-914-920, MIT Press, 2000
3. Blei, David M.; Ng, Andrew Y.; Jordan, Michael I (January 2003). "Latent Dirichlet allocation". In Lafferty, John. *Journal of Machine Learning Research* **3** (4–5): *pp.* 993–1022. doi:10.1162/jmlr.2003.3.4-5.993.
4. Smola, Alexander, and Shravan Narayanamurthy. "An architecture for parallel topic models." *Proceedings of the VLDB Endowment* 3.1-2 (2010): 703-710.
5. Zhai, Ke, et al. "Mr. LDA: A flexible large scale topic modeling package using variational inference in MapReduce." *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012