Using scores to improve language modelling of movie plot summaries

Jorge Sáez Gómez
University of Amsterdam
Amsterdam, Postbox 94216, 1090 GE
josago@gmail.com

Roelof van der Heijden
University of Amsterdam
Amsterdam, Postbox 94216, 1090 GE
roelof.heijden@gmail.com

Abstract

In this work we explore the application of Bayesian topic models on corpora of movie plot summaries. We introduce the *Supervised Latent Dirichlet Allocation* model and derive a collapsed Gibbs sampler for it, along with a couple of useful implementation hints. We created a custom movie database including language features from both plot summaries and scripts, along with movie scores. We find that using the movie scores results in slightly lower perplexities for the testing sets, and suggest future paths to further extend our work.

1 Introduction

In recent years, several successes have been booked for applying semantic analysis on user comments of movies [4, 5]. In this report we use those same techniques, but apply them to plot summaries of movies. Using these corpora of text we try to determine whether the contents of these summaries and the score these movies are rated with on the popular online movie database IMDb [2] are correlated. We do this by comparing the performance on two Latent Dirichlet Allocation models (LDA) – one with and another without using the scores. If we achieve better results when we are using the scores, we can conclude that there is a correlation between the scores and the texts of the summaries.

First we describe the characteristics of the problem and take a closer look at the data set in Section 2. Next, we outline the model and Gibbs sampler in Section 3. We describe which experiments we ran in Section 4 along with their results. Finally, we provide our final remarks in Section 5.

This project is part of the Natural Language Processing course of the UvA from Fall 2014.

2 Problem description

In this section we describe the characteristics of the problem and take a closer look at the data set that we use.

2.1 Data set

The texts that we use in ths model are summaries of movies. These summaries have been written by users of the popular online movie database IMDb, with the intent to outline the events that occur in the movie. They were gathered by crawling the IMDb website. Only movies which had their scripts available on the online script website `IMSDb.com` [3] at time of this writing were selected, to allow for an easy comparison when using the full script instead of summaries in a future study.

These corpora are fundamentally different from movie reviews, as the author is not supposed to convey his or her own opinion of the movie in the summary. However, this can obviously never be

fully prevented, since the author has seen the movie in question and is willing to spend time and effort to write the summary. In this light we make the following assumptions.

1. An important assumption we make is that the authors wrote the summaries voluntarily, without any compensation or external influence which might affect the writing of the author.
2. Additionally, we assume that the summary also contains the authors personal opinion on the movie, although this does not have to be explicitly mentioned.

Only if these assumptions hold, can we try to find a correlation between the summary and the score of the movie.

An example plot summary from IMDb can be found below. It is from the movie "Big Fish" (2003) and is written by a user who wanted to remain anonymous.

> *The story revolves around a dying father and his son, who is trying to learn more about his dad by piecing together the stories he has gathered over the years. The son winds up re-creating his father's elusive life in a series of legends and myths inspired by the few facts he knows. Through these tales, the son begins to understand his father's great feats and his great failings.*
>
> by Anonymous

However, a different summary may have different properties. Below is another plot summary, about the movie "Indiana Jones and the Last Crusade" (1989), written by IMDb user commanderblue.

> *Indiana Jones, famed adventurer and archaeologist acquires a diary that holds clues and a map with no names to find the mysterious Holy Grail – which was sent from his father, Dr. Henry Jones, in Italy. Upon hearing from a private collector, Walter Donavan, that the mission for the Holy Grail went astray with the disappearance of his father, Indiana Jones and museum curator Marcus Brody venture to Italy in search of Indy's father. However, upon retrieving Dr. Henry Jones in Nazi territory, the rescue mission turns into a race to find the Holy Grail before the Nazis do – who plan to use it for complete world domination for their super-race. With the diary as a vital key and the map with no names as a guide, Indiana Jones once again finds himself in another death defying adventure of pure excitement.*
>
> by commanderblue

Among other differences, this summary uses more proper pronouns, whose uniqueness could be difficult for our algorithm to handle properly. To deal with this, we perform some basic manipulations on the data set.

2.2 Data manipulation

To increase performance and speed up the algorithm, we prune and stem the data set.

First we prune the data set, by discarding infrequent words. We define a word to be infrequent when it only occurs in a single summary. This gets rid of words like "Donovan" from the "Indiana Jones an the Last Crusade" summary, but keeps "Indiana" and "Jones", which might have some relation with the score of the movie. It also helps to diminish the effects of overfitting.

After that, we use a basic stemmer to reduce words to their stems. This significantly reduces the number of different words that occur in the data set, which increases the execution speed of our algorithm. We use the stemmer Lingua for this project [6].

2.3 Scores and score distribution

Each movie in the data set is ranked with a certain score from IMDb. These scores lie between 1 and 10 (inclusive) and are precise up to 1 decimal position. Even if these scores are submitted by users,

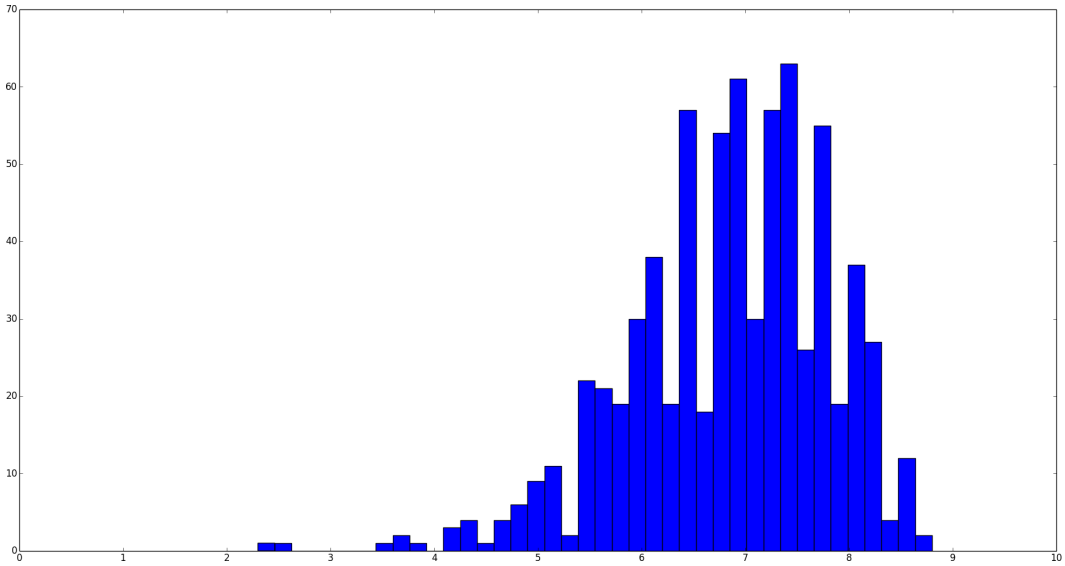

Figure 1: Score distribution between 0 and 10 of the complete data set.

IMDb still performs some normalization on them. The distribution of scores in our data set can be seen in Figure 1.

Notice how almost all scores lie between 6 and 8. This is likely due to the fact that the scripts on IMSDb are requested and provided by users of the website. So a movie that is liked by many people will have a high chance to have its script being requested and supplied on IMSDb. This results in this skewed distribution: a movie on IMSDb will have a high chance to be liked by many people.

Another thing to note is that IMDb uses a weighted average, instead of the arithmetic mean to calculate a movie's score. According to IMDb, 'various filters are applied to the raw data in order to eliminate and reduce attempts at "vote stuffing" by individuals more interested in changing the current rating of a movie than giving their true opinion of it." The exact formula that is used to calculate the weighted average is kept secret, to be able to maintain its effectiveness.

This weighted average can vary quite significantly with the arithmetic mean. For example, the arithmatic mean of the scores of the movie "The Amityville Asylum" is 3.3, while its weighted average is 2.6. IMDb claims the weighted score provides a more accurate vote average than the arithmatic mean.

Our final data set contains around 700 movies. Each one of them includes its score and the word counts for both the plot summaries and the scripts. After prunning and stemming, there are around 36000 unique words in our data set, and approximately 13 million words in total.

Table 1: Characteristics of the data set.

Total number of tokens	$12.7 \cdot 10^6$
Number of unique tokens	35000
Average number of tokens within a movie summary	75
Average number of tokens within a movie script	18000

3 Approach

In this section we explain our model, and show the derivations we use for the Gibbs sampler.

3.1 Model

In this section we describe the extended topic based model we used. It is taken from Blei & McAuliffe [1].

The generative version of LDA is as follows:

1. Draw topic proportions $\theta \mid \alpha \sim \mathrm{Dir}(\alpha)$.
2. For each word:
   (a) Draw topic assignment $z_n \mid \theta \sim \mathrm{Mult}(\theta)$.
   (b) Draw word $w_n \mid z_n, \beta_{1:K} \sim \mathrm{Mult}(\beta_{zn})$.

Its graphical representation can be seen in Figure 2. It makes use of the hyperparameters α, which influences the topic distributions of a document, and of β, which influences the topic definitions of each topic.

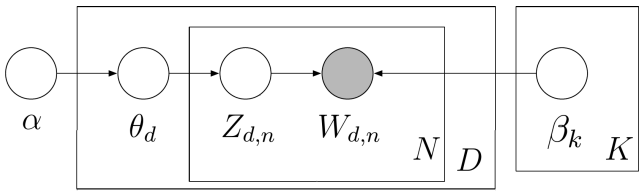

Figure 2: A graphical representation of traditional LDA model.

We however, use an extended version of LDA, which makes use of the given scores. Because of this, a third step is added to the generative process:

1. Draw topic proportions $\theta \mid \alpha \sim \mathrm{Dir}(\alpha)$.
2. For each word:
   (a) Draw topic assignment $z_n \mid \theta \sim \mathrm{Mult}(\theta)$.
   (b) Draw word $w_n \mid z_n, \beta_{1:K} \sim \mathrm{Mult}(\beta_{zn})$.
3. Draw response variable $y \mid z_{1:N}, \eta, \sigma^2 \sim \mathcal{N}(\eta^\top \bar{z}, \sigma^2)$.

This version can be called supervised LDA or SLDA. Its graphical representation can be seen in Figure 3. Note that we use a symmetric α and β, i.e. $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$, where $\forall i, j \leq K$: $\alpha_i = \alpha_j$ and similar for β.

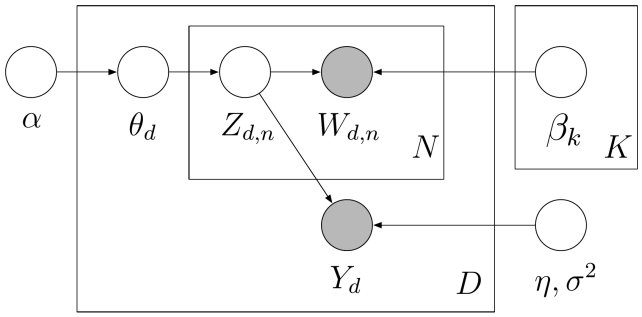

Figure 3: A graphical representation of our modified LDA model.

3.1.1 Collapsed Supervised Latent Dirichlet Allocation

In this section we will give the derivations for the collapsed supervised LDA model.

We start by recalling the full likelihood expression (i.e. for all variables, both latent and visible) of the model:

$$\begin{aligned} P(\Phi, \Theta, S, Z, W \mid \alpha, \beta, \eta, \sigma) = \left[\prod_{k=1}^{K} \mathrm{Dir}(\varphi_k \mid \beta)\right] \\ \times \left[\prod_{d=1}^{D} \mathrm{Dir}(\theta_d \mid \alpha)\mathcal{N}(s_d \mid \eta^\top \bar{z}_d, \sigma) \prod_{i=1}^{N_d} \mathrm{Mult}(z_{di} \mid \theta_d)\, \mathrm{Mult}(w_{di} \mid \varphi_{z_{di}})\right] \end{aligned} \tag{1}$$

The final collapsed SLDA will only give probabilities for S, Z and W. To achieve this, we will first integrate out the latent variable Φ. The first thing that can be noticed is that every φ_k is independently sampled, and thus can be integrated separately:

$$P(\Theta, S, Z, W \mid \alpha, \beta, \eta, \sigma) = \int_{\varphi_1} \int_{\varphi_2} \cdots \int_{\varphi_K} P(\Phi, \Theta, S, Z, W \mid \alpha, \beta, \eta, \sigma) \tag{2}$$

At this point we need to rewrite part of Equation 1 in order to be able to continue the derivation:

$$\prod_{d=1}^{D} \prod_{i=1}^{N_d} \mathrm{Mult}(w_{di} \mid \varphi_{z_{di}}) = \prod_{d=1}^{D} \prod_{w=1}^{W} \prod_{k=1}^{K} \mathrm{Mult}(w \mid \varphi_k)^{N_{dk}}, \tag{3}$$

where N_d represents the total number of words within document d, and N_{dk} represents the number of words within document d assigned to topic k.

$$\begin{aligned} P(\Theta, S, Z, W \mid \alpha, \beta, \eta, \sigma) = \left[\prod_{k=1}^{K} \int_{\varphi_k} \mathrm{Dir}(\varphi_k \mid \beta) \prod_{d=1}^{D} \prod_{w=1}^{W} \mathrm{Mult}(w \mid \varphi_k)^{N_{dk}}\right] \\ \times \left[\prod_{d=1}^{D} \mathrm{Dir}(\theta_d \mid \alpha)\mathcal{N}(s_d \mid \eta^\top \bar{z}_d, \sigma) \prod_{i=1}^{N_d} \mathrm{Mult}(z_{d,i} \mid \theta_d)\right] \end{aligned} \tag{4}$$

We can now make use of the definition of the Dirichlet-Multinomial distribution in order to solve all the integrals:

$$\begin{aligned} P(\Theta, S, Z, W \mid \alpha, \beta, \eta, \sigma) = \left[\prod_{k=1}^{K} \frac{\Gamma(W\beta)}{\Gamma(N_k + W\beta)} \prod_{w=1}^{W} \frac{\Gamma(N_{kw} + \beta)}{\Gamma(\beta)}\right] \\ \times \left[\prod_{d=1}^{D} \mathrm{Dir}(\theta_d \mid \alpha)\mathcal{N}(s_d \mid \eta^\top \bar{z}_d, \sigma) \prod_{i=1}^{N_d} \mathrm{Mult}(z_{di} \mid \theta_d)\right] \end{aligned} \tag{5}$$

We can proceed in an analogous fashion in order to integrate out the latent Θ parameter:

$$P(S, Z, W \mid \alpha, \beta, \eta, \sigma) = \int_{\theta_1} \int_{\theta_2} \cdots \int_{\theta_D} p(\Theta, S, Z, W \mid \alpha, \beta, \eta, \sigma) \tag{6}$$

This gives us:

$$\begin{aligned} P(S, Z, W \mid \alpha, \beta, \eta, \sigma) = \left[\prod_{k=1}^{K} \frac{\Gamma(W\beta)}{\Gamma(N_k + W\beta)} \prod_{w=1}^{W} \frac{\Gamma(N_{kw} + \beta)}{\Gamma(\beta)}\right] \\ \times \left[\prod_{d=1}^{D} \mathcal{N}(s_d \mid \eta^\top \bar{z}_d, \sigma) \frac{\Gamma(K\alpha)}{\Gamma(N_d + K\alpha)} \prod_{k=1}^{K} \frac{\Gamma(N_{dk} + \alpha)}{\Gamma(\alpha)}\right] \end{aligned} \tag{7}$$

Note that $\Gamma(\beta)$ and $\Gamma(\alpha)$ are constant in these products, so they can be moved outside of the product.

$$\begin{aligned} P(S, Z, W \mid \alpha, \beta, \eta, \sigma) = \left[\prod_{k=1}^{K} \frac{\Gamma(W\beta)}{\Gamma(\beta)^W \Gamma(N_k + W\beta)} \prod_{w=1}^{W} \Gamma(N_{kw} + \beta)\right] \\ \times \left[\prod_{d=1}^{D} \mathcal{N}(s_d \mid \eta^\top \bar{z}_d, \sigma) \frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K \Gamma(N_d + K\alpha)} \prod_{k=1}^{K} \Gamma(N_{dk} + \alpha)\right] \end{aligned} \tag{8}$$

3.1.2 Collapsed Gibbs Sampler

Using Bayes' theorem, we know that:

$$P(Z \mid S, W, \dots) = \frac{P(S, W \mid Z, \dots)P(Z \mid \dots)}{P(S, W \mid \dots)} \propto P(S, W, Z \mid \dots), \tag{9}$$

where we did not write all model hyperparameters for the sake of clarity. Thus, in order to implement a Gibbs sampler for this model, we have that:

$$P(z_{di} = k \mid Z^{\setminus i}, S, W, \alpha, \beta, \eta, \sigma) \propto P(z_{di} = k, Z^{\setminus i}, S, W, \alpha, \beta, \eta, \sigma) =$$

$$\left[\prod_{k'=1}^{K} \frac{\Gamma(W\beta)}{\Gamma(\beta)^W \Gamma(N_k'^{\setminus i} + \mathbb{I}(k' = k) + W\beta)} \prod_{w=1}^{W} \Gamma(N_{k'w}^{\setminus i} + \mathbb{I}(k' = k \wedge w = w_{di}) + \beta)\right]$$

$$\times \left[\prod_{d=1}^{D} \mathcal{N}\left(s_d \;\middle|\; \eta^T \frac{N_{dk'}^{\setminus i} + \mathbb{I}(k' = k)}{N_d}, \sigma\right) \frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K \Gamma(N_d^{\setminus i} + K\alpha)} \prod_{k'=1}^{K} \Gamma(N_{dk'}^{\setminus i} + \mathbb{I}(k' = k) + \alpha)\right]$$

Where we used $\frac{N_{dk}}{N_d} \equiv \bar{z_d}$ and $\mathbb{I}(\text{condition}) = 1$ iff the condition evaluates to true, and zero otherwise. We can further simplify the previous expression by removing those factors that are constant across all possible values for k, resulting in the following final collapsed Gibbs sampler:

$$\begin{aligned} P(z_{di} = k \mid Z^{\setminus i}, S, W, \dots) \propto & \left[\prod_{k'} \frac{\prod_w \Gamma(N_{k'w}^{\setminus i} + \mathbb{I}(k' = k \wedge w = w_{di}) + \beta)}{\Gamma(N_{k'}^{\setminus i} + \mathbb{I}(k' = k) + W\beta)}\right] \\ & \times \mathcal{N}\left(s_d \;\middle|\; \eta^T \frac{N_{dk'}^{\setminus i} + \mathbb{I}(k' = k)}{N_d}, \sigma\right) \prod_{k'} \Gamma(N_{dk'}^{\setminus i} + \mathbb{I}(k' = k) + \alpha), \end{aligned} \tag{10}$$

The Gibbs sampler can also be expressed in log-space probabilities, in order to get a more numerically-stable implementation:

$$\begin{aligned} & \log P(z_{di} = k \mid Z^{\setminus i}, S, W, \alpha, \beta, \eta, \sigma) \\ & \propto \sum_{k', w} \left[\log \Gamma(N_{k'w}^{\setminus i} + \mathbb{I}(k' = k \wedge w = w_{di}) + \beta) - \log \Gamma(N_{k'}^{\setminus i} + \mathbb{I}(k' = k) + W\beta)\right] \\ & - \frac{1}{2\sigma^2}\left(s_d - \eta^\top \frac{N_{dk'}^{\setminus i} + \mathbb{I}(k' = k)}{N_d}\right)^2 + \sum_{k'} \log \Gamma(N_{dk'}^{\setminus i} + \mathbb{I}(k' = k) + \alpha) \end{aligned}$$

3.1.3 Rewriting the log-gamma function

The logarithm of the gamma function can be rewritten as follows [7]:

$$\log \Gamma(z) = -\gamma z - \log z + \sum_{j=1}^{\infty} \left[\frac{z}{j} - \log\left(1 + \frac{z}{j}\right)\right], \tag{11}$$

where γ is the Euler-Mascheroni constant. We apply this to $\sum_k \sum_w \log \Gamma(N_{kw} + \beta)$, which then becomes:

$$\begin{aligned} & = \sum_{k,w} -\gamma(N_{kw} + \beta) - \log(N_{kw} + \beta) + \sum_{j=1}^{\infty} \frac{N_{kw} + \beta}{j} - \log\left(1 + \frac{N_{kw} + \beta}{j}\right) \\ & = -\gamma(N + KW\beta) - \sum_{k,w} \log(N_{kw} + \beta) - \sum_{j=1}^{\infty} \frac{N_{kw} + \beta}{j} - \log\left(\frac{N_{kw} + \beta + j}{j}\right) \end{aligned}$$

Note that the term $-\gamma(N + KW\beta)$ serves as a normalisation constant for this data set. Since we do not need the exact probabilities but only the proportional probabilities during the algorithms execution, we can discard those terms.

$$
\begin{aligned}
&\Rightarrow -\sum_{k,w} \log(N_{kw} + \beta) - \sum_{j=1}^{\infty} \frac{N_{kw} + \beta}{j} - \log\left(\frac{N_{kw} + \beta + j}{j}\right) \\
&= -\sum_{k,w} \log(N_{kw} + \beta) - \sum_{j=1}^{\infty} \frac{N_{kw} + \beta}{j} - \log(N_{kw} + \beta + j) + \log(j) \\
&= -\sum_{k,w} \log(N_{kw} + \beta) - \sum_{j=1}^{\infty} \left(\frac{N_{kw} + \beta}{j} + \log(j)\right) + \sum_{j=1}^{\infty} \log(N_{kw} + \beta + j) \\
&= \sum_{j=1}^{\infty} \left(\frac{N + KW\beta}{j} + KW\log(j)\right) - \sum_{k,w} \log(N_{kw} + \beta) + \sum_{j=1}^{\infty} \log(N_{kw} + \beta + j) \\
&= \sum_{j=1}^{\infty} \left(\frac{N + KW\beta}{j} + KW\log(j)\right) - \sum_{k,w} \sum_{j=0}^{\infty} \log(N_{kw} + \beta + j)
\end{aligned}
$$

Again, the term $\sum_{j=1}^{\infty} \frac{N+KW\beta}{j} + KW\log(j)$ is a constant for this data set, so we can discard it. This results in the following proportionality:

$$\sum_{k,w} \log\Gamma(N_{kw} + \beta) \propto -\sum_{k,w} \sum_{j=0}^{\infty} \log(N_{kw} + \beta + j) \tag{12}$$

Using similar steps, we can also simplify

$$\sum_{k} \log\Gamma(N_{k} + W\beta) \propto -\sum_{k} \sum_{j=0}^{\infty} \log(N_{k} + W\beta + j) \tag{13}$$

$$\sum_{k,d} \log\Gamma(N_{dk} + \alpha) \propto -\sum_{k,d} \sum_{j=0}^{\infty} \log(N_{dk} + \alpha + j) \tag{14}$$

$$\sum_{d} \log\Gamma(N_{d} + K\alpha) \propto -\sum_{d} \sum_{j=0}^{\infty} \log(N_{d} + K\alpha + j) \tag{15}$$

Putting these together gives us the following proportionality:

$$
\begin{aligned}
&\log P(z_{di} = k \mid Z^{\setminus i}, S, W, \alpha, \beta, \eta, \sigma) \\
&\propto \sum_{j=0}^{\infty} \sum_{k'} \log(N_{k'}^{\setminus i} + \mathbb{I}(k' = k) + W\beta + j) - \log(N_{dk'}^{\setminus i} + \mathbb{I}(k' = k) + \alpha + j) \\
&- \sum_{j=0}^{\infty} \sum_{k',w} \log(N_{k'w}^{\setminus i} + \mathbb{I}(k' = k \wedge w = w_{di}) + \beta + j) - \frac{1}{2\sigma^2}\left(s_d - \eta^\top \frac{N_{dk'}^{\setminus i} + \mathbb{I}(k' = k)}{N_d}\right)^2
\end{aligned}
$$

In practice, however, we can not calculate the infinite terms of the above sums, and thus we would stop the approximation at some arbitrary iteration J:

$$
\begin{aligned}
&\log P(z_{di} = k \mid Z^{\setminus i}, S, W, \alpha, \beta, \eta, \sigma) \\
&\propto -\frac{1}{2\sigma^2}\left(s_d - \eta^\top \frac{N_{dk'}^{\setminus i} + \mathbb{I}(k' = k)}{N_d}\right)^2 + \sum_{k'} \sum_{j=0}^{J} \log(N_{k'}^{\setminus i} + \mathbb{I}(k' = k) + W\beta + j) \\
&- \log(N_{dk'}^{\setminus i} + \mathbb{I}(k' = k) + \alpha + j) + \sum_{w} \log(N_{k'w}^{\setminus i} + \mathbb{I}(k' = k \wedge w = w_{di}) + \beta + j)
\end{aligned}
$$

3.1.4 Estimating response parameters

The maximum a posteriori (MAP) estimate for the hyperparameter η can be inferred from the gradient of the complete model likelihood:

$$\nabla_{\eta_k} \log P(S, Z, W \mid \alpha, \beta, \eta, \sigma) = \sum_d \frac{1}{\sigma^2} \frac{N_{dk}}{N_d} \left(s_d - \eta^T \frac{N_{d\cdot}}{N_d} \right) \tag{16}$$

$$= \sum_d \frac{s_d \frac{N_{dk}}{N_d}}{\sigma^2} - \sum_d \frac{\frac{N_{dk}}{N_d} \left(\eta^T \frac{N_{d\cdot}}{N_d} \right)}{\sigma^2}$$

For the MAP estimate, the gradient should be zero. This allows us to rewrite the above formula into:

$$\sum_d s_d \frac{N_{dk}}{N_d} = \sum_d \frac{N_{dk}}{N_d} \left(\sum_{k'} \eta_{k'} \frac{N_{dk'}}{N_d} \right)$$

$$\sum_d s_d \frac{N_{dk}}{N_d} = \sum_d \frac{N_{dk}}{N_d} \left(\eta_k \frac{N_{dk}}{N_d} + \sum_{k' \neq k} \eta_{k'} \frac{N_{dk'}}{N_d} \right)$$

$$\sum_d s_d \frac{N_{dk}}{N_d} = \eta_k \sum_d \left(\frac{N_{dk}}{N_d} \right)^2 + \sum_d \left(\frac{N_{dk}}{N_d} \sum_{k' \neq k} \eta_{k'} \frac{N_{dk'}}{N_d} \right)$$

Further rewriting finally gives us the MAP estimate for η:

$$\eta_k \sum_d \left(\frac{N_{dk}}{N_d} \right)^2 = \sum_d \left(s_d \frac{N_{dk}}{N_d} - \frac{N_{dk}}{N_d} \sum_{k' \neq k} \eta_{k'} \frac{N_{dk'}}{N_d} \right)$$

$$\eta_k \sum_d \left(\frac{N_{dk}}{N_d} \right)^2 = \sum_d \frac{N_{dk}}{N_d} \left(s_d - \sum_{k' \neq k} \eta_{k'} \frac{N_{dk'}}{N_d} \right)$$

$$\eta_k = \frac{\sum_d \frac{N_{dk}}{N_d} \left(s_d - \sum_{k' \neq k} \eta_{k'} \frac{N_{dk'}}{N_d} \right)}{\sum_d \left(\frac{N_{dk}}{N_d} \right)^2} \tag{17}$$

From our experiments, we noticed that trying to apply the formula as described in Equation 17 as an update rule for η does not converge. We theorize that this is due to the fact that every element within η is dependent on the values for all other elements. Instead, the following update can be used:

$$\eta_k^{new} \leftarrow (1 - \gamma) \eta_k^{old} + \gamma \frac{\sum_d \frac{N_{dk}}{N_d} \left(s_d - \sum_{k' \neq k} \eta_{k'} \frac{N_{dk'}}{N_d} \right)}{\sum_d \left(\frac{N_{dk}}{N_d} \right)^2 + \varepsilon}, \tag{18}$$

where we used $1 \gg \gamma > 0$ in order for the previous series to converge and $1 \gg \varepsilon > 0$ as a smoothing constant that helps avoiding infinities in the previous formula.

4 Experiments

For our experiments we chose to use two different performance measures: perplexity and inverse accuracy.

4.1 Perplexity

The perplexity measure is commonly used within LDA literature [8], and we include it for reference purposes . It is defined as follows:

$$\text{Perplexity} \equiv \exp \left\{ -\frac{1}{N} \sum_{i=1}^{N} \log P(x_i) \right\}, \tag{19}$$

where $P(x_i)$ represents the probability the model gives for the i-th item within the test set. The lower the perplexity, the less "surprised" the model is of seeing its input, which indicates a better model.

This measure has a shortcoming, though. It is enough for the model to give a probability of zero to a word to drive this performance measure to infinity. This does not mean, however, that the model is infinitely bad, since the rest of the items might still have large probabilities. To overcome this problem, we also measure the performance in terms of the average inverse accuracy of the model.

4.2 Inverse accuracy

The average inverse accuracy of the model is defined as follows:

$$\text{Accuracy}^{-1} \equiv \frac{1}{\frac{1}{N}\sum_{i=1}^{N} P(x_i)} \tag{20}$$

This measure should be interpreted as the number of words the model incorrectly predicts for each correctly predicted word. The lower this magnitude, the better the model.

4.3 Results

We created a first experiment in order to check the validity of our approximation for the Gibbs sampler of our model. We selected different values for J and compared the predictive performance for each case. We ran three different MCMC chains and then averaged the results. Within each execution, 100 movies were randomly selected as the training set, and 20 as the testing set. We used 5 samples, taken every 4 steps of the Gibbs sampler, with the first 20 iterations discarded as the burn-in period. Our findings are summarized in Table 2:

Table 2: Experiment results for varying values of J with 100/20 movies as training/test set.

J	1	2	3	4	5	6
Perplexity	4100	4125	4082	4251	3754	3881
Accuracy^{-1}	858	871	823	863	877	815

We can observe that the predictive performance tends to improve with larger values of J. This is to be expected, since the higher J is, the better the approximation of the log-gamma function becomes. This improvement is not, however, very consistent even when averaging across 3 chains, and thus we theorize that the extra computational time spent on making J higher is better used if J is kept as low as possible and more movies and/or iterations are used instead. We then conclude this approximation is interesting, at least from a computational perspective.

We devised a second experiment in order to find out good values for the number of topics to use in our models. Again we did an equivalent set up to the one for the previous experiment, but this time changing the number of topics and whether or not the Gibbs sampler uses the movie score information.

Table 3: Experiment results for varying values of K, with $J = 1$ and 100/20 movies as training/test set.

Metric	LDA topics		25	50	100
Perplexity	Using scores?	No	4128	5333	6954
		Yes	**4005**	5503	7082
Accuracy^{-1}	Using scores?	No	845	1318	2049
		Yes	**805**	1372	2067

We can see how all models start overfitting with, at least, more than 25 topics. This is most likely due to the fact that we only used 100 movie plot summaries for training, which is a rather small data set. We also observe that, for 25 topics, not only the best predictive performance is achieved, but

this is further improved by using the movie scores. This improvement is very small, but we could systematically observe it in all the tests we run for this work.

We also wondered which words were the most correlated with either bad or good movie scores. In order to discover this, the hyperparameter η can be multiplied with the topic definitions Φ, and the resulting vector ordered by values. Table 4 shows a list with the 30 words with the highest positive correlation with good movie scores. We noticed that a vast majority of words share the minimum score (0.00137), which suggests that most words do not convey any movie score information.

Table 4: An ordered list with the 30 words with the highest positive correlation with good movie scores. $K = 25$, $J = 1$ and 300 movies as training set.

Word	Score	Word	Score
his	2.86	get	0.500
he	2.34	up	0.471
(empty)	1.64	have	0.466
her	1.48	life	0.461
who	1.14	out	0.449
she	0.882	after	0.444
they	0.870	man	0.406
their	0.843	take	0.401
him	0.795	this	0.388
has	0.788	new	0.357
be	0.766	becom	0.356
when	0.750	world	0.351
from	0.749	into	0.346
find	0.541	time	0.346
one	0.516	them	0.329

Note how these words are very general in nature, and many can likely be found in every summary. Some exceptions to this rule are present, such as the words "world" and "becom", the latter being the result of stemming words like "becoming" and "becomes". We speculate that movies in which a character undergoes a series of changes, finally becoming a different person than he used to be, or where the entire world is the stage of the movie, are more likely to become high rated movies.

But why do common words like "he" or "who" dominate the list? Another hypothesis is that summaries which talk a lot about what the protagonist(s) do, tend to correlate with high rated movies.

Also noteworthy is the fact that male-oriented words appear higher in the list than female oriented words. Apparently, both "his" and "he" affect the score in a significantly bigger way than "hers" and "she" do. Maybe this is a reflection of the fact that in many modern movies men feature more prominently than women, or a reflection of the audience of the movies in the data set: maybe men like to watch movies which center around men rather than movies that center around women.

Finally, we noticed when compiling this list that there are no words that have a negative effect on the score. This is probably another occurence of the "long tail" phenomenon which is frequently seen in the area of NLP: most words have almost no effect on the score. The fact that there are no words that correlate negatively is surprising, though. This might be because the writers of the summaries are not supposed to inject their own opinion of the movie in the summary, but rather more objectively depict what occurs in the movie. Because of this, negative words could simply not occur in the data set.

Now we re-run the experiment with the best settings so far with several times more movies, and these results are shown in table 5. This experiment confirms several facts: that our implementation indeed works, that there is still room for predictive improvement if more data is used, and that the using movie scores is indeed marginally better.

Table 5: Experiment results with $J = 1$ and 300/60 movies as training/test set. Four MCMC chains were run and their results averaged.

Metric	LDA topics		25
Perplexity	Using scores?	No	3037
		Yes	**2960**
Accuracy^{-1}	Using scores?	No	561
		Yes	**549**

5 Discussion

In this project we set out to see if the words that are used in plot summaries are correlated to the score of the movie. We did this by comparing two LDA models, one which does and one which does not use the score of the movie in its prediction.

Although the performance of both versions is similar, the algorithm performs better when it does use the scores. The algorithm tends to overfit very quickly, as the best performance was observed when only 25 topics were used, compared to 50 or 100 topics. In this case, a perplexity score of 4005 and an inverse accuracy of 805 was achieved, with 100/20 movies as training/test set.

The words which influence the prediction of the algorithm the most are very common and can likely be found in every summary. Also, male oriented words score higher than female oriented words, possibly because many modern movies center around men or because men watch more movies than women and score movies about men higher as well.

And finally, there are no words which correlate negatively with the score of the movie. This can be attributed to the fact the summaries are meant to be a neutral depiction of the events in the movie. Words with a negative connotation could simply not occur in the data set.

While working on this project, we noticed that the algorithm is very slow, and as a result can only handle small data sets. A larger data set could increase performance, but before that can realistically implemented one first needs to speed up the algorithm. This could potentially be done making use of GPUs.

If a sufficient speed up is achieved, one could also look at full movie scripts as input text, rather than plot summaries.

In conclusion, we can say that using scores is an improvement to the algorithm, but the summaries of movies still form a text corpus that is too big to be handled effectively with our current implementation.

Acknowledgements

We would like to acknowledge Dr. Ivan Titov, the lecturer of the NLP class of Fall 2014, and his PhD student and teaching assistant Ehsan Khoddam Mohammadi, who helped us take the first steps in this project.

Work distribution

In this section we will outline the distribution of the work that was done for this project, as required by the assignment.

Jorge Sáez Gómez did most of the coding, experimenting and mathematical derivations, with help from Roelof van der Heijden. The latter wrote the most of this report, with help from Jorge where needed. The presentation was written by both of them.

Francesco Stablum, who was initially also a member of this group, did not make significant contributions to this project.

References

[1] David M. Blei, Jon D. McAuliffe; *Supervised Topic Models*; Neural Information Processing Systems 21 (2007)

[2] IMDb.com, Inc.; *IMDb, the world's most popular and authorative source for movie, TV and celebrity content*; `www.imdb.com`

[3] IMSDB.com; *Internet Movie Script Database - Movie scripts free for reading and downloading*; `www.imdsb.com`

[4] M. Joshi, D. Das, K. Gimpel, N. A. Smith; *Movie Reviews and Revenues: An Experiment in Text Regression*; In Proceedings of NAACL-HLT (2010)

[5] A. T. Scaria, R. M. philip, S. V. Mehta; *Predicting Star Ratings of Movie Review Comments*; `http://cs229.stanford.edu/proj2011/MehtaPhilipScaria-Predicting%20Star%20Ratings%20from%20Movie%20Review%20Comments.pdf`

[6] Jim Richardson, Benjamin Franz; *Lingua Stem*; Freerun Technologies, Inc; (1999); `http://search.cpan.org/~snowhare/Lingua-Stem-0.84/lib/Lingua/Stem.pod`

[7] Geogre Boros, Victor H. Moll, *Irresistible Integrals: symbolics, analysis and experiments in the evaluation of integrals*; Cambridge University Press (2004)

[8] Arthur Asuncion, Max Welling et al.; *On smoothing and inference for topic models*; In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (2009)