Statistical clustering of text documents with PLSA

Taneli Saastamoinen

1st of March 2016

Contents

Chapter 1

Introduction

This thesis is about probabilistic latent semantic analysis (PLSA), which is a method for clustering text documents. The historical context of PLSA and other topic models is briefly summarised. The mathematical basis of PLSA is explained, as is the practical algorithm for fitting the PLSA model. PLSA is also compared and contrasted with other techniques of text clustering. Finally some practical results of applying PLSA to English Wikipedia documents are discussed.

Chapter 2

Topic models

Topic models have been a subject of research since the late 1980s as part of the larger context of information retrieval. Information retrieval refers to algorithms and methods used to analyse and index documents, usually ones consisting of natural-language text [14, 17]. The goal is to be able to build search engines with which a large number of documents can efficiently be queried to find, say, documents containing certain keywords, or those that are about a certain topic. This latter goal is the one attempted by topic models. It is fundamentally important that the methods used are not only automatable and accurate but also efficient, so that they can be applied to very large collections of documents.

Information retrieval is a rather expansive, cross-discipline topic of study. It encompasses elements from lingustics and language technology as well as mathematics, statistics and computer science. This thesis restricts itself to topic models, specifically the PLSA topic model.

The goal of topic models is to cluster the given documents by their topic. The resulting output is, depending on the method used, either a weighted list of topics for each given document, or a method to compute the "topic distance" between each document that can then be used for clustering.

Topic models are an *unsupervised* technique, i.e. the model is not given a set of "correctly" classified articles and topics to start with; rather the topic model creates its own topics from scratch, so to speak, and classifies the articles into these topics as best it can. The resulting topics and classifications are not necessarily the same ones a human would come up with, and therefore the results may reveal connections and differences that were not noticed before. On the other hand, interpreting the results may be difficult.

It is important to note that the topic models described in this thesis do not take into account the order of the words inside the documents. Rather, they belong to the category of *bag of words* models. The idea is that it is not necessary to examine the writing style

of a document or the context of individual words to determine the topic of the document; just the word frequencies are sufficient.

When using a topic model in practice, the given natural-language documents are usually first preprocessed using suitable language technology techniques [14]. An example is removing stop words and then stemming the remaining words. In what follows it is assumed that any such preprocessing has already been done.

Chapter 3

LSA

An early topic model is latent semantic analysis (LSA) [5, 6], a predecessor of PLSA. LSA builds on the earlier *vector space model* [8], which is a way to represent the documents and words of the material. Thanks to this representation, techniques of ordinary linear algebra can be used to handle the calculations.

After preprocessing, the documents in the given corpus are stored in an $n \times m$ matrix A, where n is the number of documents and m the number of distinct words in the corpus. Each row of A corresponds to a document and each column to a word, such that the element A_{ij} is the number of occurrences of the word j in the document i. The distance between any two documents is now easy to determine with the vector space model. The distance is defined as the angle between the corresponding document vectors, which is straightforward to compute with the dot product:

$$a \cdot b = \sum_{j=0}^{m} a_j b_j = \|a\| \, \|b\| \, \cos\theta \;\Leftrightarrow\; \theta = \cos^{-1}\left(\frac{a \cdot b}{\|a\|\|b\|}\right),$$

where a and b are document vectors and θ the angle between them. The distance between any two words in the corpus can be computed in a similar way.

Thinking further about the properties of the vector space model, it is easy to see see, for instance, that if one takes an arbitrary document x and forms a new document y by concatenating two copies of x, then the angle between the corresponding document vectors of x and y is zero. This makes intuitive sense, since the relative word frequencies of both documents are the same, and thus the documents are in that sense identical. This result is also easily shown mathematically, proceeding directly from the definition of the

dot product:

$$\begin{aligned}
\cos\theta &= \frac{x \cdot y}{\|x\|\|y\|} \\
&= \frac{\sum_j x_j y_j}{(\sqrt{\sum_j x_j^2})(\sqrt{\sum_j y_j^2})} \\
&= \frac{\sum_j x_j 2x_j}{(\sqrt{\sum_j x_j^2})(\sqrt{\sum_j (2x_j)^2})} \\
&= \frac{2\sum_j x_j^2}{(\sqrt{\sum_j x_j^2})(2\sqrt{\sum_j x_j^2})} \\
&= \frac{2\sum_j x_j^2}{2\sum_j x_j^2} \\
&= 1.
\end{aligned}$$

Similarly, by pondering the differences between documents in the vector space model, it is seen that the angle between two document vectors grows as does the difference between the word frequencies of the documents, and the largest difference is caused by words that only appear in one of the two documents. As an extreme example, if x is a document in which at least one of the words, say w_q, does not appear at all, and if y is a document that only contains the word w_q, then the angle between the document vectors x and y is 90 degrees. This is again easy to show mathematically by direct calculation:

$$\begin{aligned}
\cos\theta &= \frac{x \cdot y}{\|x\|\|y\|} \\
&= \frac{\sum_j x_j y_j}{(\sqrt{\sum_j x_j^2})(\sqrt{\sum_j y_j^2})} \\
&= \frac{x_q y_q + \sum_{j \neq q} x_j y_j}{(\sqrt{\sum_j x_j^2})(\sqrt{\sum_j y_j^2})} \\
&= \frac{0}{(\sqrt{\sum_j x_j^2})(\sqrt{\sum_j y_j^2})} \\
&= 0.
\end{aligned}$$

The vector space model then is in this sense a rather practical and intuitive-seeming way to describe documents and words. One problem however is that in many natural

corpora the number of unique words in the corpus is very large. Since in addition most documents only contain a small fraction of all possible words, the document vectors end up being very large (i.e. high-dimensional) and also sparse. Therefore handling natural-language text is not quite painless. In practice the cardinality of the word set, and thus the dimensionality of the document vectors, can reach the tens of thousands.

The fundamental idea of LSA is that the word-document matrix A can be treated in the same way as any other matrix, and therefore standard linear algebra techniques such as *singular value decomposition* (SVD) can be used. Once the SVD of A is available, it can be used to significantly reduce the dimension of A by retaining only the k most significant singular values and discarding the rest. With this well-known method the approximation $\tilde{A}$ can be obtained:

$$A = U\Sigma V' \rightarrow \tilde{A} = U_k \Sigma_k V_k',$$

where $U\Sigma V'$ is the singular value decomposition of A. Here Σ_k is a $k \times k$ diagonal matrix containing only the k largest singular values on its diagonal, and the matrices U_k and V_k', of size $m \times k$ and $k \times n$ respectively, contain only the rows and columns corresponding to these k singular values.

As is known, the matrix $\tilde{A}$ is the best possible approximation for the original A in the sense of the smallest square error [18]. LSA proceeds to argue that it can therefore be reasonably claimed that $\tilde{A}$ contains only the most meaningful part of the original data, the "signal" – the noise having been removed.

Determining a suitable value for k is not trivial. As with other clustering methods, one can perform a search for a suitable k by seeing what proportion of the variance in the data is explained by forming the approximation $\tilde{A}$ with the first k' singular values. When the data consists of natural text, the value of k thus obtained will tend to be quite large; a handful of singular values will simply not be enough, due to the characteristics of natural language. For example, the authors of LSA used a k value of 100 [6]. Even though $k = 100$ is somewhat large, their data consisted of 1000-1500 documents, so the reduction in dimensionality is significant.

Once a suitable k is chosen and the singular value decomposition performed, the resulting matrices U_k, V_k and Σ_k are much smaller in total size than the original word-document matrix, which makes practical computation noticeably easier. With these matrices the original words and documents can be projected into the k-dimensional space produced by the singular value decomposition. This space is called the latent semantic space. Intuitively, the meaning or semantics of the documents can be deduced from their word frequencies; thanks to the singular value decomposition we now know the most important characteristics of these frequencies, and thus the hidden, or latent, semantics become visible.

The projection into the latent semantic space can be easily done, for both documents

and words, by using the matrices obtained from the singular value decomposition. For instance the matrix $\tilde{A}\tilde{A}'$ contains each dot product between documents in the latent semantic space, and

$$\begin{aligned}\tilde{A}\tilde{A}' &= (U_k\Sigma_k V_k')(U_k\Sigma_k V_k')' \\ &= U_k\Sigma_k' V_k' V_k \Sigma_k' U_k' \\ &= U_k\Sigma_k\Sigma_k U_k' \\ &= U_k\Sigma_k^2 U_k',\end{aligned}$$

because V_k is an orthonormal matrix and Σ_k a diagonal matrix. From this it is seen that a dot product between documents i and j is equal to the dot product of the rows i and j of the matrix $U_k\Sigma_k$. By a similar argument, the dot products between the words are obtained from the matrix $V_k\Sigma_k$.

In addition to investigating the distances between documents and words, LSA can also be used to implement a search engine. Given a seach query, i.e. a list of words, these words are simply treated in the same way as the actual documents. First the query words are preprocessed (by removing stop words, stemming etc) in the same way as the documents. Then the query pseudo-document is projected into the latent semantic space, after which finding documents similar to the query can be done by calculating dot products as described above. In addition, the results can be ordered by distance, that is, similarity.

3.1 LSA's strengths and weaknesses

In practice LSA is straightforward to implement and the results are fairly good. A fundamental part is played by singular value decomposition. Its calculation is a well-studied problem, and several efficient approximation methods exist that have acceptable accuracy as well [6]. The dimensionality reduction achieved with the singular value decomposition also means that the original documents can be compressed into a relatively small space, i.e. described with k coordinates, where $k \ll m, n$. Therefore the method can efficiently handle large numbers of documents.

A good feature of LSA is its handling of synonyms. In natural language, synonyms, i.e. different words for the same concept, occur with some frequency. Even though LSA treats words only as character strings, it can nevertheless "understand" synonyms, as long as the given material is comprehensive enough. To see this, consider a document d_1 in which some concept is only referred to with the word a, and another document d_2 in which only a different word b is used for this concept. In many natural corpora there will also be other documents d_s in which both a and b occur together (e.g. survey articles). If this is the case, the synonyms a and b can be connected to each other, because the documents d_s

are close to both d_1 and d_2 as can be readily seen by examining the relevant dot products. Therefore, when answering a search query, an LSA-based search engine can return not only the documents that contain the given terms, but also documents that contain only their synonyms, which is a very useful property for a search engine.

Polysemy, however, cannot be handled properly by LSA. Polysemy refers to the same word, that is the same sequence of characters, having two distinct meanings. An example would be the word "can". Because in LSA's vector space model each word corresponds to precisely one point in the vector space, a word simply cannot be in more than one cluster at the same time. A polysemic word will therefore end up in some arbitrary point between the clusters it actually belongs to.

A greater shortcoming than polysemy is the LSA model itself. Singular value decomposition, as a least-squares method, is most applicable for normally distributed data [13]. However word frequencies in natural text are not in fact normally distributed, following rather the Poisson distribution or the Zipf distribution. Although fairly good results can in practice be obtained with the singular value decomposition, handling documents in this way is rather ad hoc; a solid mathematical foundation for such a model has not been proposed.

One weakness of topic models in general is their handling of words one at a time. In many cases some amount of context would be useful, for instance in English, where set phrases such as "morning walk" are written as two separate words. (In some other languages, such as German and Finnish, one-word compounds are used instead.) Another example is place names such as Los Gatos, Los Angeles etc, where the common word "los" will tend to be more common than any of the full phrases, which is unfortunate since it conveys less information. This problem is common to all "bag of words" models and is not specific to LSA. One solution would be to use n-grams instead of simple words [1].

Chapter 4

PLSA

Probabilistic latent semantic analysis (PLSA) is a topic model published by Thomas Hofmann in 1999 [10, 11]. As seen from the name, the idea of PLSA is to be a probabilistic version of LSA. Although the problem under consideration is identical, the probabilistic starting point leads to a model completely different from LSA.

The starting point of PLSA is to use a latent variable model [3] to describe the documents and topics. Intuitively, it is assumed that there are hidden variables, topics, which determine the meanings of the given words and documents. These topics cannot be observed directly, but must rather be estimated based on the visible variables – the documents and words.

Each topic might then be a collection of words, or, even better, a probability distribution on the words. In a similar vein each document could be associated with one or more topics, i.e. each document could have a probability distribution over the topics.

More precisely, call the documents d_n, the words w_m and the topics z_k, where $n = 1, \ldots, N$ and N is the number of documents; similarly let M be the number of words and K the number of topics. The first thing to do is to select an appropriate K. As with LSA, the analysis can be performed for various candidate values K' and the value giving the best results chosen. Usually this results in values of K similar to those used with LSA. Hofmann uses the value $K = 128$.

The model building begins from the idea that the given documents are "generated" by starting from the topics (latent variables) and proceeding from them to the words and documents (visible variables). One important assumption the model makes is the naive Bayes assumption, which is common in latent variable models; i.e. it is assumed that the data points are independent of each other given the latent variable.

A suitable distribution for the topics might be the categorical distribution, which is a special case of the multinomial distribution. After drawing a topic z from the topic distribution $P(z)$, the next step is to draw a word and a document from the distributions

$P(w|z)$ and $P(d|z)$ respectively. For these the multinomial distribution is a natural choice. The naive Bayes assumption can be stated as

$$P(d|z)P(w|z)P(z) = P(d, w|z)P(z)$$

and from this is obtained the joint distribution of the documents and words by marginalising over the topics:

$$\begin{aligned} P(d, w) &= \sum_z P(d, w|z)P(z) \\ &= \sum_z P(d|z)P(w|z)P(z). \end{aligned}$$

The goal of the model fitting then is to estimate $P(z)$, $P(w|z)$ and $P(d|z)$, given $P(d, w)$.

A different model equivalent to this one can be obtained by first choosing a document, then the topic and then the word. Later it turns out that such a model is more handy in practical calculations. Therefore in what follows this alternate model is used. The equivalence is easily seen by direct calculation:

$$\begin{aligned} P(d, w) &= \sum_z P(d, w|z)P(z) \\ &= \sum_z P(d|z)P(w|z)P(z) \\ &= \sum_z \frac{P(z|d)P(d)}{P(z)} P(w|z)P(z) \\ &= \sum_z P(z|d)P(d)P(w|z) \\ &= P(d) \sum_z P(z|d)P(w|z). \end{aligned}$$

In this model the distributions to estimate are $P(d)$, $P(z|d)$ and $P(w|z)$, given $P(d, w)$.

As is apparent from the model equations, the PLSA model is a mixture model, specifically a mixture of multinomials. For such a mixture model, ordinary maximum likelihood estimation cannot be used directly, because the model is too complicated. Instead the *EM algorithm*, or expectation maximisation algorithm [7], is used for parameter estimation. The EM algorithm is kind of a two-phase iterated version of the maximum likelihood method. It is commonly used e.g. for fitting Gaussian mixture models. Another important special case of the EM algorithm is k-means. These are both extremely useful clustering methods in their own right.

The EM algorithm does not attempt to directly maximise the likelihood function or its logarithm, but instead the target of maximisation is the log-likelihood function's expected value with respect to the posterior of the hidden variables, that is, the expected value

$$\mathcal{Q} \equiv \mathbb{E}_z \left[\sum_x \log \mathcal{L} \right] = \sum_x \sum_z P(z|x,\theta) \log \mathcal{L}.$$

It can be shown that maximising $\mathcal{Q}$ is equivalent to maximising the original log-likelihood function. A detailed derivation is shown in appendix A.

The EM algorithm maximises $\mathcal{Q}$ by iterating two steps. In the *E step* the posterior of the hidden variables, $P(z|x,\theta)$, is calculated, treating any other variables as given, i.e. as constants. The *M step* involves maximisation of $\mathcal{Q}$ itself, treating the posterior of the hidden variables as given. The latter step can often be accomplished using ordinary maximum likelihood methods.

The EM algorithm can be applied to the PLSA model as follows. Let $n(d,w)$ denote the total number of occurrences of word w in document d. The total likelihood function of PLSA can be written as (since constants can be ignored)

$$\mathcal{L} = \prod_d \prod_w P(d,w)^{n(d,w)}$$

and from this it is easy to derive the expected value of the log-likelihood function as required for the EM algorithm:

$$\begin{aligned}
\mathcal{Q} &= \mathbb{E}_z \left[\sum_d \sum_w n(d,w) \log P(d,w) \right] \\
&= \sum_d \sum_w n(d,w) \sum_z P(z|d,w) \log P(d,w) \\
&= \sum_d \sum_w n(d,w) \sum_z P(z|d,w) \log \left[P(d) P(z|d) P(w|z) \right] \\
&= \sum_d \sum_w n(d,w) \sum_z P(z|d,w) \left[\log P(d) + \log P(z|d) + \log P(w|z) \right].
\end{aligned}$$

Applying the EM algorithm is now straightforward:

E step: compute the posterior of the hidden variables by direct calculation:

$$\begin{aligned} P(z|d,w) &= \frac{P(z,d,w)}{P(d,w)} \\ &= \frac{P(d)P(z|d)P(w|z)}{\sum_z P(d)P(z|d)P(w|z)} \\ &= \frac{P(z|d)P(w|z)}{\sum_z P(z|d)P(w|z)}. \end{aligned} \tag{4.1}$$

M step: maximise $\mathcal{Q}$, the expected value of the log likelihood function, treating the posterior of the hidden variables as given. The method used here is ordinary maximum likelihood estimation. First, Lagrange multipliers are added to ensure that the probability distributions to be maximised remain valid, i.e. will sum to unity after maximisation. After this the equation is differentiated with respect to each distribution in turn and the result set to zero:

$$\begin{aligned} U = &\sum_d \sum_w n(d,w) \sum_z P(z|d,w) \left[\log P(d) + \log P(z|d) + \log P(w|z)\right] \\ &+ \alpha(1 - \sum_d P(d)) + \sum_z \beta_z (1 - \sum_w P(w|z)) + \sum_d \gamma_d (1 - \sum_z P(z|d)). \end{aligned}$$

For example

$$\frac{\partial U}{\partial P(d)} = \sum_w n(d,w) \sum_z \frac{P(z|d,w)}{P(d)} - \alpha.$$

The maximisation finally results in

$$\begin{aligned} P(d) &= \frac{\sum_w n(d,w) \sum_z P(z|d,w)}{\sum_d \sum_w n(d,w) \sum_z P(z|d,w)} \\ &= \frac{\sum_w n(d,w)}{\sum_d \sum_w n(d,w)}, \end{aligned} \tag{4.2}$$

$$\begin{aligned} P(z|d) &= \frac{\sum_w n(d,w) P(z|d,w)}{\sum_z \sum_w n(d,w) P(z|d,w)} \\ &= \frac{\sum_w n(d,w) P(z|d,w)}{\sum_w n(d,w)}, \end{aligned} \tag{4.3}$$

$$P(w|z) = \frac{\sum_d n(d,w) P(z|d,w)}{\sum_w \sum_d n(d,w) P(z|d,w)}. \tag{4.4}$$

This completes the derivation of the EM algorithm. The estimate for $P(d)$ is easy to calculate immediately and does not change over the course of the iteration. $P(z|d)$ and

$P(w|z)$ can be initialised from e.g. the uniform distribution and are then updated on each iteration. After initialisation, the E and M steps are alternated until the parameter estimate no longer changes.

The algorithm's results are the fitted distribution tables of $P(z|d)$ and $P(w|z)$. Based on these, estimates can now be made about which documents are related to which topics and which topics are related to which words, just as desired. It is also possible to calculate the mutual distances between documents, between words and between topics, and with these the documents can be clustered and indexed.

It is not obvious how new, unseen documents should be treated after the model has been fitted. Handling of such new documents is fundamentally important, not only because of the desired search engine application but also, for example, to enable cross-validation. With LSA any new document could easily be projected into the smaller-dimensional space specified by the model, but with PLSA the equivalent operation is more involved. Hofmann [11] proposes that new documents be "folded in" to the existing model by performing a partial EM iteration, in which the parameters $P(w|z)$ are kept constant and only the weights $P(z|d')$ of the new document d' are estimated. In practice this works rather well and enables both cross-validation and search engine features for the model.

One problem to be aware of with PLSA is overfitting. To avoid overfitting, the usual method of cross-validation can be used [11]. A suitable stopping criterion for cross-validation is *perplexity*, which is commonly used for natural text analysis:

$$\mathcal{P} = \exp\left[-\frac{\sum_{d'}\sum_{w'} n(d', w') \log P(w'|d')}{\sum_{d'}\sum_{w'} n(d', w')}\right]. \tag{4.5}$$

Here d' and w' denote the documents and their words, respectively, that were left out of the model fitting to be used in the test set instead, and the probability $P(w'|d')$ is calculated from the fitted model after performing the folding in as described above.

From the definition of perplexity it is seen that if the model assigns a very small probability $P(w'|d')$ to some new document or word, then the negative logarithm of this small probability is very large, which results in a large contribution to the overall perplexity. On the other hand if the model assigns some reasonably large probability to each document and word it has seen, then the perplexity will not be overly large. The larger the model's predicted probability is for each unseen document and word, on average, the smaller the perplexity.

One practical problem with perplexity is that since natural text corpora tend to have a very high number of unique words, it may happen that the documents used for fitting the model do not include every word in the material, and the perplexity of such unknown words is by definition infinite. This problem can be circumvented by assigning some small probability to such completely unseen words or by simply ignoring them when calculating the perplexity.

In addition to cross-validation, many other useful techniques can be used with PLSA to help with model fitting. For example, Hofmann mentions that he used inverse annealing to improve the rate of convergence of the EM algorithm. Many such improvements of the EM algorithm have been proposed and investigated, but they are outside the scope of this thesis.

4.1 PLSA's strenghts and weaknesses

One definite advantage of PLSA is that it is easy to implement. As the distributions involved are discrete, the numerical calculations required are straightforward. The posterior estimate and the maximum likelihood estimates can be computed with a few simple loops, as is readily seen from the EM algoritm equations 4.1 – 4.4. The results obtained with PLSA are also fairly good and according to Hofmann better than LSA's results for the same input.

Probably the most significant problem of PLSA is that its model consists of a number of parameters linear in the size of the corpus. A distribution table is calculated for each word and each document, and the size of these tables inevitably grows linearly with the number of words and documents. This leads to the calculation being heavier and slower as the number of documents increases, and also to inevitable overfit: eventually there are so many parameters in the model that overfitting is all but guaranteed. This problem cannot be completely fixed with cross-validation or other such methods, since the issue arises from the structure of the model.

It is difficult to say how large the data set can be before PLSA's overfit becomes a real problem. In practice this has to be investigated on a case-by-case basis. In general it can be said that PLSA gives good results as long as the size of the corpus is not overly large. Despite the overfitting problem PLSA is a valid technique, but when using it one has to be aware of the unavoidable limitations presented by the overfitting.

Chapter 5

Advanced topic models

5.1 LDA

Latent Dirichlet allocation (LDA) is a topic model proposed by Blei, Ng and Jordan [4]. It can be seen as an improved version of PLSA. The distribution tables produced by PLSA naturally raise the question of whether it would not be better to represent the algorithm's results as fitted parameters for some suitable distribution, from which the tables were "generated". This reasoning leads to the LDA model.

According to LDA, each document is thought to have been generated as follows. First the document length $M \sim \text{Poisson}(\xi)$ is generated and a parameter vector $\theta \sim \text{Dir}(\alpha)$ is drawn from some suitable Dirichlet distribution. Each word w_m in the document is then generated by first drawing, for each word separately, a topic z_m from the multinomial $\text{Mult}(\theta)$, after which the word itself is drawn from the word distribution $p(w_m|z_m, \beta)$. The parameter β represents the weights between the words and the topics.

As is seen, the LDA model is similar to PLSA but richer. The Dirichlet distribution is known to be the conjugate prior of the multinomial, and with its use LDA is rid of the multinomial distribution tables of PLSA and the problems associated with them, namely model size and overfitting. PLSA is in fact a special case of LDA: PLSA produces a point estimate whereas LDA fits the entire distribution. As Blei at al write, PLSA cannot model the "space between documents", unlike LDA.

The LDA model however is complicated enough that it cannot be calculated directly. In practice numerical methods, such as Markov chain Monte Carlo or variational approximation, must be used to find the posterior of the hidden variables.

After the LDA model has been fitted with some suitable method, its results are better than those obtained with PLSA, according to Blei et al [4]. LDA therefore is a genuine improvement over PLSA.

5.2 Other topic models

Both PLSA and LDA have been extensively investigated and improved on since they were originally published. One recent result is additive regularisation as proposed by Vorontsov and Potapenko [19]. According to them, PLSA's accuracy and efficiency can be significantly improved by regularising the topics and the document-topic weights. Regularisation here refers to adding certain contraints to the model equations in order to try to steer the model-fitting algorithm away from ill-defined or undesirable states. Vorontsov and Potapenko present several possible regularisation constraints and show that with them the simple, easy-to-fit PLSA model can be made to also be as rich as the LDA model or even richer.

Topic models are an active topic of research, and new results and approaches continue to be published.

Chapter 6

Practical experiments with PLSA

I also tried PLSA in practice. I wrote a simple PLSA implementation in Scala and used it to examine articles of the online encyclopedia Wikipedia. I used the English Wikipedia, since for the English language there are plenty of good tools available for the required preprocessing.

I first went to the Wikipedia archives page [20] and downloaded all articles of English Wikipedia in XML format. Of these I discarded the articles with no content, such as articles containing only a redirect and articles which had no category assigned. 6471454 articles remained.

First I drew a random sample of 596 articles. I preprocessed these by removing any technical markup and stop words and stemming the remaining words with the improved version of Porter's English stemmer [15, 16]. In addition I removed words that only occur in one article in the sample; clearly such words cannot be helpful when fitting the model. Finally there were 7931 words left of the original 24019, which is about one third.

After preprocessing I built the word-document matrix and gave it to my PLSA implementation. I used 30 topics, i.e. 30 latent variables. The stopping criterion was perplexity with ten-fold cross-validation (that is, k-fold cross-validation with $k = 10$). I did not utilise annealing or other advanced methods, but did the iteration directly with the original EM algorithm equations (4.1 – 4.4).

As can be predicted, such a small sample from such a large and wide data set probably will not result in a very well-fitting model. By examining the fitted model, I indeed saw that a large part of the 30 topics are located between the documents. Some topics however are representative. The table 6.1 details the most important words of three best-fitting topics. By a fitting topic I refer here to a topic which, in the fitted model, can by itself carry the majority of the weight of each of its documents. Similarly a less fitting topic is one which is mostly associated with documents whose topic distribution consists of several different topics, each with a small weight.

Table 6.1: The most fitting topics. Data: 596 random Wikipedia articles.

topic 1	topic 2	topic 3
film	footbal	album
archiv	player	record
australia	nation	year
inform	team	play
book	season	track
languag	american	state
nation	san	releas
univers	play	first
publish	new	mclean
american	game	magazin

The most fitting topics are fairly understandable. The first topic seems to describe culture, more specifically movies and novels. In this sample the works seem to mostly originate from Australia and the USA, which seems plausible, since the data set after all is from the English-language Wikipedia. The word "inform" seems to be less relevant to this topic.

The second topic is clearly football, perhaps American football. The word "san" is probably an artifact of the sample; it has most likely originally been a part of some set phrase such as San Francisco, San Diego etc, and ended up separate from them in the preprocessing. The third topic is clearly popular music.

Based on these retrieved topics it can be said that PLSA can be useful even when applied to a fairly small and scattered sample. Even though most topics and article-topic weights are rather nondescriptive, there are nevertheless some topics which fit very well and are therefore helpful for understanding the data and determining the categories of the documents.

Next I drew a different sample from the Wikipedia data to see how well the topic model works when the topics are easier to tell apart, i.e. when the data is less scattered. Wikipedia reports, for almost all of its articles, the categories to which article belongs; these categories are designated by humans and are therefore presumably very accurate. I therefore chose arbitrarily a few interesting categories and drew from the data set articles that belong to these categories according to the Wikipedia classification. In addition I also included articles of any other category, to provide some "noise". For fitting the model I continued to use only the actual text of the articles, not the human-designated categories.

I chose the categories "mathematics", "statistics", "biology", "literature" and "enter-

tainment". In the data set there were in all 54416 articles that, according to the human-made Wikipedia classification, belonged to one or more of these categories. The remaining 6417038 articles belonged to other categories. The sample was 600 articles, of which 235 belonged to the desired categories and the other 365 belonged to other categories. As before, I used ten-fold cross-validation and 30 topics.

Table 6.2: The most fitting topics. Data: 600 Wikipedia articles (see text).

topic 1	topic 2	topic 3	topic 4
game	literatur	census	anim
version	publish	age	seri
video	born	famili	manga
releas	writer	popul	magazin
fight	die	household	state
ninja	english	town	one
final	poet	u	right
charact	novel	spring	will
feature	book	femal	charact
lin	john	watch	year

This time, of the 30 topics of the model, half were somewhat fitting and four were very fitting. The most fitting ones are seen in table 6.2. As expected, the model found all five selected categories, although some of the topics associated with these five were less fitting than others and some of the five categories corresponded to more than one topic. The most fitting topics for the five categories were video games (a form of entertainment), literature, and anime and manga (another form of entertainment). The most fitting other topics were, somewhat surprisingly, cities and regions, and after that again football and popular music. As before, even the best-fitting topics have a small number of artifacts, but not enough to hurt their understandability.

Based on these experiments it can be said that even the most ordinary PLSA model is fairly good at classifying arbitrary text articles according to their topics and at determining the most common topics occurring in the data set. The model works better with data sets that have a clearer separation of topics and that are not too scattered topic-wise. Indeed, the model can also be used to investigate how scattered the topics of the data set are: if there are very many distinct topics compared to the size of the data set, the model does not fit very well. In this case to get a better result the simplest way would be to increase the sample size.

Finally it must be noted that writing your own implementation of PLSA or other

similar algorithms is not necessary these days. There are various free and open libraries available that provide implementations of LSA, PLSA, LDA and their variants for many commonly used statistical programs, such as Spark MLlib [2] and R [9]. A simple PLSA implementation is however easy enough to write from scratch, and this can provide more insight into how the PLSA model works.

Bibliography

[1] Abedi, V., Yeasin, M., Zand, R. "Empirical study using network of semantically related associations in bridging the knowledge gap". *Journal of Translational Medicine* 12 (2014): 324.

[2] Apache Software Foundation. "Spark MLLib". `https://spark.apache.org/docs/latest/mllib-guide.html`. 2016.

[3] Bartholomew, D.J., Knott, M., Moustaki, I. *Latent Variable Models and Factor Analysis: A Unified Approach*, 3rd Edition. John Wiley & Sons, 2011.

[4] Blei, D.M., Ng, A.Y., Jordan, M.I. "Latent Dirichlet allocation." *The Journal of machine Learning research* 3 (2003): 993-1022.

[5] Deerwester, S.C., et al. "Using latent semantic analysis to improve access to textual information." *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1988.

[6] Deerwester, S.C., et al. "Indexing by latent semantic analysis." *JAsIs* 41.6 (1990): 391-407.

[7] Dempster, A.P., Laird, N.M., Rubin, D.B. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1. (1977): 1-38.

[8] Dubin, David. "The Most Influential Paper Gerard Salton Never Wrote." *Library Trends* 52(4), Spring 2004: 748-764.

[9] Free Software Foundation. "The R Project for Statistical Computing". `https://www.r-project.org/`. 2016.

[10] Hofmann, Thomas. "Probabilistic latent semantic indexing." *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999.

[11] Hofmann, Thomas. "Unsupervised learning by probabilistic latent semantic analysis." *Machine learning* 42.1-2 (2001): 177-196.

[12] Kullback, S., Leibler, R.A. "On information and sufficiency". *Annals of Mathematical Statistics* 22.1 (1951): 79–86.

[13] Manning, C.D., Schütze, H. *Foundations of Statistical Natural Language Processing*, pp. 565-566. MIT Press 1999.

[14] Manning, C.D., et al. *Introduction to Information Retrieval*. Cambridge University Press 2008.

[15] Porter, M.F. "An algorithm for suffix stripping". *Program*, 14.3 (1980): 130-137.

[16] Porter, M.F. "The English (Porter2) stemming algorithm". `http://snowball.tartarus.org/algorithms/english/stemmer.html`. 2001-2002.

[17] Rijsbergen, van, C.J. *Information Retrieval*, 2nd Ed. Butterworth-Heinemann 1979.

[18] Trefethen, L.N., Bau, D. *Numerical Linear Algebra*, pp. 35-36. Society for Industrial and Applied Mathematics 1997.

[19] Vorontsov, K., Potapenko, A. "Additive regularization of topic models". *Machine Learning*, 101.1-3(2015), 303-323.

[20] Wikimedia Foundation. "Wikimedia Downloads". `https://dumps.wikimedia.org/`. 2016.

[21] Wu, C. F. J. "On the Convergence Properties of the EM Algorithm". *Annals of Statistics* 11.1 (1983): 95–103.

Appendix A

The EM algorithm

The expectation maximisation (EM) algorithm [7] is a technique most commonly used to fit mixture models that contain hidden variables. Suitable distributions for these models include especially the exponential family of distributions, such as the normal distribution and the multinomial distribution, and also other "well-behaved" distributions.

The object of interest is a mixture model whose hidden variables z indicate which component each data point x is drawn from. Such a model's log likelihood function is

$$\begin{aligned}\log \mathcal{L} &= \sum_x \log P(x|\theta) \\ &= \sum_x \log \sum_z P(x|z,\theta)P(z|\theta).\end{aligned}$$

Due to the sum of the logarithm in the equation, maximising this directly is difficult. Therefore a different path of investigation is taken. A useful result to utilise is *Jensen's inequality*: when f is a concave function (such as the logarithm),

$$f(\mathbb{E}[X]) \geq \mathbb{E}[f(X)].$$

To start with, multiply and divide the log likelihood function by the arbitrary distri-

bution $q(z)$, after which Jensen's inequality is applied:

$$\begin{aligned}
\log \mathcal{L} &= \sum_x \log \sum_z P(x|z,\theta)P(z|\theta) \\
&= \sum_x \log \sum_z P(x|z,\theta)P(z|\theta)\frac{q(z)}{q(z)} \\
&= \sum_x \log \mathbb{E}_q \left[\frac{P(x|z,\theta)P(z|\theta)}{q(z)}\right] \\
&\geq \sum_x \mathbb{E}_q \left[\log \frac{P(x|z,\theta)P(z|\theta)}{q(z)}\right] \\
&= \sum_x \sum_z q(z) \log \frac{P(x|z,\theta)P(z|\theta)}{q(z)}. \qquad \text{(A.1)}
\end{aligned}$$

The form A.1 can be further developed as follows:

$$\begin{aligned}
\log \mathcal{L} &= \sum_x \log P(x|\theta) \\
&\geq \sum_x \sum_z q(z) \log P(x|z,\theta)P(z|\theta)\frac{1}{q(z)} \qquad \text{(A.2)} \\
&= \sum_x \sum_z q(z) \log \frac{P(x,z,\theta)}{P(z,\theta)}\frac{P(z,\theta)}{P(\theta)}\frac{1}{q(z)} \\
&= \sum_x \sum_z q(z) \log \frac{P(x|\theta)P(\theta)P(z|x,\theta)}{P(z,\theta)}\frac{P(z,\theta)}{P(\theta)}\frac{1}{q(z)} \\
&= \sum_x \sum_z q(z) \log \frac{P(x|\theta)P(z|x,\theta)}{q(z)} \\
&= \sum_x \left[\sum_z q(z) \log P(x|\theta) + \sum_z q(z) \log \frac{P(z|x,\theta)}{q(z)}\right] \\
&= \sum_x \left[\log P(x|\theta) - \sum_z q(z) \log \frac{q(z)}{P(z|x,\theta)}\right] \\
&= \sum_x \left[\log P(x|\theta) - D_{KL}[q(z)||P(z|x,\theta)]\right], \qquad \text{(A.3)}
\end{aligned}$$

where D_{KL} is the Kullback-Leibler divergence, which is always non-negative [12]. From this it is seen that A.2 holds with equality, as long as the K-L divergence from $q(z)$ to $P(z|x,\theta)$ is zero. This is the case iff $q(z) = P(z|x,\theta)$.

Therefore let us set $q(z) \equiv P(z|x,\theta)$. Computing this is the *E step* of the EM algorithm: given the data x and some kind of estimate for the parameters θ, it is possible to calculate an estimate for the posterior distribution of the hidden variables, $P(z|x,\theta)$, and this estimate maximises A.3.

Now in A.1, perform the substitution of $P(z|x,\theta)$ for q and develop further:

$$\begin{aligned}
\log \mathcal{L} &= \sum_x \log P(x|\theta) \\
&= \sum_x \log \sum_z P(x|z,\theta) P(z|\theta) \\
&\geq \sum_x \sum_z P(z|x,\theta) \log \frac{P(x|z,\theta)P(z|\theta)}{P(z|x,\theta)} \\
&= \sum_x \sum_z P(z|x,\theta) \log \frac{P(x|z,\theta)P(z|\theta)P(x,\theta)}{P(z,x,\theta)} \\
&= \sum_x \sum_z P(z|x,\theta) \log \frac{P(x|z,\theta)P(z|\theta)P(x,\theta)}{P(x|z,\theta)P(z|\theta)P(\theta)} \\
&= \sum_x \sum_z P(z|x,\theta) \log \frac{P(x,\theta)}{P(\theta)} \\
&= \sum_x \sum_z P(z|x,\theta) \log P(x|\theta) \\
&= \mathbb{E}_z \sum_x \log P(x|\theta). \qquad \text{(A.4)}
\end{aligned}$$

The form A.4 is often denoted by $\mathcal{Q}$ in the literature. From A.3 and the substitution $q(z) \equiv P(z|x,\theta)$ it follows that if a maximum can be found for $\mathcal{Q}$, this will also be the maximum of the original log likelihood function $\log \mathcal{L}$.

Thus we have arrived at the *M step* of the EM algorithm. If we have an estimate of the posterior of the latent variables, $P(z|x,\theta)$, we can treat it as a constant, after which maximising $\mathcal{Q}$ is often straightforward. In the M step this maximisation is performed using some suitable technique, such as maximum likelihood estimation.

The complete EM algorithm is as follows:

1. Initialise the parameter estimate θ randomly, for instance from a uniform distribution.

2. *E step*: calculate the posterior of the latent variables, $P(z|x,\theta)$, based on the data x and the parameter estimate θ.

3. *M step*: calculate the parameter estimate θ with e.g. maximum likelihood estimation, using the posterior $P(z|x, \theta)$.

4. Repeat the E and M steps until the parameter estimate converges.

The details of how to compute the parameter estimate in the M step depend on the application. With many distributions, this can be done with ordinary maximum likelihood estimation.

The proof of the convergence of the EM algorithm [21] is omitted.