TOPIC MODELLING FOR SUPPORTING SYSTEMATIC REVIEWS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2016

By
Yuanhan MO
School of Computer Science

Contents

List of Tables

List of Figures

Abstract

Identifying relevant studies for inclusion in a systematic review (i.e. screening) is a complex, laborious and expensive task. Recently, a number of studies have shown that the use of machine learning and text mining methods to automatically identify relevant studies have the potential to drastically decrease the workload involved in the screening phase. The vast majority of available machine learning methods exploit the same underlying principle, i.e. a study is modelled as a bag-of-words (BOW).

This thesis explores the use of topic modelling methods to derive a more informative representation of studies. Latent Dirichlet Allocation (LDA) is applied, an unsupervised topic-modelling approach, to identify topics from a collection of studies. Then each study is represented as a distribution of LDA-topics. Additionally, Topics derived by LDA are enriched with technical multi-word terms identified by an automatic term recognition (ATR) tool. For experimentation, SVM-based classifiers are applied using either the topic-based or the BOW representation to automatically identify relevant studies.

The results obtained show that the SVM classifier is able to identify more relevant studies when using the LDA representation than the BOW representation. Moreover, this study demonstrates that kernel functions used in SVM obtain a superior performance when using LDA feature representations. These observations hold for two systematic reviews of the clinical domain and three reviews of the social science domain.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I wish to thank my supervisor Prof. Sophia Ananiadou for providing me undeserved support and encouragement throughout the research. I also wish to thank Dr. Georgios Kontonatsios, for his patient listening, and useful contribution. He also helped to proof read what was not his subject.

Chapter 1

Introduction

As data grows in volume, systematic review has begun to play an increasingly important role in many areas, especially for evidence-based medicine (EBM). Systematic review can provide researchers with high quality evidence for a pre-defined question. However, the process of developing a systematic review is traditionally performed manually, and the need for its automation is increasingly important.

In order to obtain high-quality evidence relevant to a research topic in EBM, hundreds of documents from the biomedical literature must be manually examined by reviewers during screening, the most tedious and burdensome phase in systematic reviews. Recently, a number of studies have shown that the use of machine learning and text mining to automatically identify relevant studies can drastically reduce the workload involved in the screening phase.

Automatic or semi-automatic text classification is the most usual means to assist the screening phase. Researchers have already explored the performance of many classifiers. However, the majority of available classification methods exploit the same underlying principle, i.e. a document is modelled as a bag-of-words (BOW).

Topic models, especially Latent Dirichlet Allocation (LDA) [BNJ03], are techniques which extract the latent information from an observation such as text or image. Applying topic models to the citations found during systematic reviews enables the automatic screening phase to achieve a better performance.

1.1 Systematic Reviews

A systematic review aims to summarise evidence for a pre-defined research question; it has been widely applied in the biomedical and healthcare contexts. A typical systematic review takes the following steps: 1) a thorough search for the relevant literature in appropriate databases, citation indexes and journals. 2) two independent reviewers manually assess the titles and abstracts according to a pre-defined protocol for eligibility and relevance. The *Cochrane Collaboration* provides eight-step guidelines for building a systematic review [HGA+11]:

1. Defining the review question and developing criteria for including studies.
2. Searching for studies.
3. Selecting studies and collecting data.
4. Assessing risk of bias in included studies.
5. Analysing data and undertaking meta-analyses.
6. Addressing reporting biases.
7. Presenting results and "summary of findings" tables.
8. Interpreting results and drawing conclusions.

Systematic reviews are not only powerful tools for producing medical evidence, but they are also used in other sciences such as software engineering, social media and so on.

1.1.1 Screening Phase in Systematic Reviews

A traditional screening phase (namely, Step 5 in the eight-step guidelines) for systematic reviews is to manually identify and assess citations relevant to a research topic, according to a certain pre-defined protocol [BTL09, Cou97, BNB+10, DFL07] known as the **PICO** framework, which seeks to identify the **P**opulation, the **I**ntervention, the **C**omparison and the **O**utcome. Manual screening means that reviewers need to read hundreds of citations during the screening phase, due to the exponential growth of biomedical literature [HC06]. However, manual screening is relatively expensive and time-consuming. According to [WSBT10], an experienced reviewer is able to

screen two abstracts per minute on average, with more complex abstracts taking longer. Moreover, a reviewer needs to identify all eligible studies (i.e. 95%-100% recall) [Coh11, MTOEA14] in order to minimise publication bias. Another issue is that the number of relevant citations is usually significantly lower than the number of irrelevant ones, which means that reviewers are dealing with an extremely unbalanced dataset. Thus, researchers are beginnning to apply machine learning, text mining [ARO+09, OETM+15], text classification [AACZ14], active learning [WTL+10] and so on, to automate this process and maintain the quality of the review as far as possible.

1.1.2 Automate the Screening Phase in Systematic Reviews

So far, many approaches based on machine learning have been shown to be helpful in reducing the workload of the screening phase [OETM+15], with impressive results. The majority of reported methods exploit automatic or semi-automatic text classification to assist the screening phase. Text classification is normally performed using the BOW model where words in documents are regarded as an unordered input.

1.2 Text Classification Basis

A number of synonyms describe the text classification task including document categorisation, topic identification and document routing [GLWW00]. Manual text classification is an expensive and time-consuming task. However, automated text classification aims to assign a document to one or more known classes by trained machines (classifiers). Usually, these classifiers can be constructed automatically by machine learning, statistical pattern recognition, or neural network approaches. A classic application of this task is spam filtering which tries to distinguish spam emails from ordinary messages.

Text classification can be done in two different ways: content-based and request-based classification [Soe85]. **Content-based classification** focuses on using the subjects in documents to determine which classes the document belongs to. These subjects can be weighted, for example, in automatic text classification high-frequency words in a document may have a bigger impact on how this document is classified. **Request-based classification** receives the requests from users, and these requests influence the classification task.

As automatic text classification is regarded as an application of machine learning on

text, it can be divided into three different processes: **Supervised text classification** requires external information (human feedback) to teach the classifiers what kind of characteristics a class has. **Unsupervised text classification** (also known as clustering) can perform the task without external information. For example, uncategorised documents that have been represented in vector space can be automatically classified into different groups by knowing the parameter k (how many classes are in these documents). **Semi-supervised text classification** is a more intelligent approach than supervised text classification as only parts of the documents need to be labelled by humans. The BOW model is the most popular model in nature language processing, where each document in the collection can be transformed into a machine readable feature vector. For example, the short sentence "you know I know you know" will be the same as "know know know you you I" and {know:3, you:2, I:1} in text classification, because the order of words is no longer important to machines, even if the latter one has completely lost the meaning of the original sentence. Classifiers only pay attention to statistical features which become valuable when there are hundreds of documents.
A typical automatic text classification task is usually constructed by the following steps:

1. **Feature selection:** This process determines the candidate features (via tokenization) and refines them using statistics or information theory. The result of the process is a dictionary which covers all useful tokens that appear in a corpus.

2. **Document encoding:** This process will produce a representation of each document, normally in feature vectors; each element in feature vectors denotes how important a token is to classification (i.e. its weight).

3. **Model training:** Classifiers receive the documents with labels (the name or identifier of a class) as input for training. Normally, only supervised learning needs a training process in order to adapt the model.

4. **Evaluation:** There is a variety of measures to evaluate the performance of classifiers. Accuracy is the most common way for text classification. However, the appropriateness of measures may vary under different circumstances. For example, accuracy may not be a suitable indicator of performance when the dataset is extremely unbalanced.

1.2.1 Feature Selection

Feature selection is the most fundamental part of applications for nature language processing, determining the features used for representation of text. In most cases, tokenization breaks a given sequence of text into pieces using the same standard, which is the first step to any further operations. A common model for tokenization is ***n*-gram**, where a contiguous sequence of n elements is developed from the given sequence of text. The elements of n-gram can be many things such as words, letters or syllables depending on the application. An n-gram model with $n = 1$ is referred to as a "unigram" or "1-gram", size 2 is a "bigram" or "2-gram". Table 1.1 shows an example of applications of using n-gram models with different n parameter.
Tokens in documents are merged and the redundant tokens should be removed in order to obtain a candidate feature list [RU12]. The size of the list is defined as dimensionality, which refers to the number of features the document has. The candidate feature list contains a large number of stop words that do not contribute to the classification performance, such as "is", "and", "are" and so on, because words like these appear abundantly in a corpus, and they do not contribute to the distinctiveness of documents. Thus, the list must be refined for a better-quality dictionary (vocabulary). Existing methods to remove stop words include locating these words by using a pre-defined stop-word list, many statistical approaches are also able to deal with this, for example, mutual information and information gain. It is worth noting that text classification may be performed for a specific domain so a customised stop-word list is necessary for that domain.

1.2.2 Document Encoding

Document encoding makes use of statistical techniques in order to transform a document into a feature vector which can define a point in the n-dimensional Euclidean space, where n denotes the size of the dictionary. An ordinary dictionary would contain more than 1,000 words after pre-processing (stop-word removal, stemming). The documents represented as points in a vector space are not visible to humans. However, with such high dimensions, the patterns among these points can be easily discovered by machines within a reasonable time. The encoding process produces a feature vector $< x^{(1)}, x^{(2)}, ..., x^{(n)} >$ for each document, where $x^{(i)}$ stands for the weight of each feature. There are many ways to calculate the weights for documents with different points

Table 1.1: Examples of *n*-gram

Applications	Elements	Samples	uni-gram	2-gram	3-gram
Text	words	I know you know I know	I, know, you, know,...	I know, know you,...	I know you, know you I,...
Text	letters	I know you know I know	I, _, k, n, o, w, ,...	I_, _k, kn,...	I_k, _kn, kno,...
DNA	base pair	ATGCGCA	A, T, G, C,...	AT, TG, GC,...	ATG, TGC, GCG,...

of focus. The most common schemes include Term Frequency (TF), Term Frequency/Inverse Document Frequency (TF/IDF) and Binary Code. It is hard to say which can outperform the others, as it always depends on the practical situation and the experience of researchers; and sometimes a prior comparison is demanded to determine the best solution for encoding documents.

1.2.3 Model Training

Training a model is the most important part for supervised learning and semi-supervised learning. In the field of machine learning, statistical learning is gaining in popularity especially in fields like computer vision, natural language processing and sport events [SC14]. Statistical learning infers an optimal function from labelled training data [MRT12] and makes a robust prediction for test data. Three major principles for statistical learning are: **Model**, **Strategy** and **Algorithm** [Han12].

1. Model

What kind of model will be trained is the first issue the researchers need to consider. A trained model receives an input X represented in feature space and produces a prediction Y. Let lower-case letters of input and output variables be the values of X and Y. So an input instance x can be written as in Eq. 1.1

$$x_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(j)}, ..., x_i^{(n)}) \tag{1.1}$$

where $x_i^{(j)}$ denotes the j-th feature in the i-th instance. So the training data is as given in Eq 1.2

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{1.2}$$

During supervised learning, the target model may be the potential conditional probability functions or decision functions. A hypothesis space $\mathcal{F}$ can be defined to include all possible functions. For decision functions:

$$\mathcal{F} = \{f | Y = f(X)\} \tag{1.3}$$

Here $\mathcal{F}$ is a family of functions determined by a parameter vector θ

$$\mathcal{F} = \{f | Y = f_\theta(X), \theta \in \mathbf{R}^n\} \tag{1.4}$$

where θ is defined on an n-dimensional Euclidean space $\mathbf{R}^n$.
Similarly, the hypothesis space for conditional probability functions with parameters θ is

$$\mathcal{F} = \{P|P_\theta(Y|X), \theta \in \mathbf{R}^n\} \tag{1.5}$$

Inferring the optimal parameters is the purpose of the training.

2. Strategy

Hypothesis space $\mathcal{F}$ defines all possible conditional probability functions or decision functions. Statistical learning aims to identify the optimal model from this space according to a certain strategy. Thus, a mechanism needs to be designed to indicate whether a model produces a good prediction or a bad one.

2.1 Loss Function and Risk Function

Given a sample space $\mathcal{X} \in \mathbf{R}^n$ and a label space $\mathcal{Y} \in \mathbf{R}$. A random variable X can be defined as an input taking values in $\mathcal{X}$. The function $f(X)$ yields a random variable Y as an output taking values in $\mathcal{Y}$. We can define a loss function or cost function $L(Y, f(X))$ to measure the errors.
Common loss functions that are widely used in statistical learning include:

1. **0-1 loss function**

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases} \tag{1.6}$$

2. **Quadratic loss function**

$$L(Y, f(X)) = (Y - f(X))^2 \tag{1.7}$$

3. **Absolute loss function**

$$L(Y, f(X)) = |Y - f(X)| \tag{1.8}$$

4. **Log-likelihood loss function**

$$L(Y, P(Y|X)) = -\log P(Y|X) \tag{1.9}$$

The smaller the value of the loss function is, the better will be the model. The input and output (X,Y) are two random variables generated by the joint probability distribution $P(X,Y)$. Thus, the expectation of the loss function is

$$R_{exp}(f) = E_p[L(Y,f(X))] = \int_{\mathcal{X}\times\mathcal{Y}} L(y,f(x))P(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{1.10}$$

This is the theoretical loss of model $f(X)$, which is called the risk function or expected loss [Han12]. The aim is a model with the minimum expected loss. It should be realized that the joint probability distribution $P(X,Y)$ is usually unknown. Consequently, it is necessary to utilize a training set independently and identically distributed samples drawn from the sample space for minimizing empirical risk.

$$T = \{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\} \tag{1.11}$$

Empirical risk R_{emp} is a more realistic measure to indicate the mean loss of a model trained with the sample of size N (training data set).

$$R_{emp}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i,f(x_i)) \tag{1.12}$$

The difference between the expected risk $R_{exp}(f)$ and the empirical risk $R_{emp}(f)$ is that $R_{exp}(f)$ indicates the expected loss of a model at the joint probability distribution. $R_{emp}(f)$ indicates the mean loss at the sample. R_{emp} will become more and more concentrated around R_{exp} if the sample size is large enough ($n \to \infty$). Thus, it is intuitive to use the empirical risk as a proxy to estimate the expected risk. However, the training sample is finite in the realistic situation, so the estimation of R_{exp} using R_{emp} is sometimes not as good as expected unless it is corrected by some means: empirical risk minimisation or structural risk minimisation.

2.2 Empirical Risk Minimisation and Structural Risk Minimisation

Empirical risk minimisation (ERM) considers a model with the minimum empirical risk to be the optimal model. According to this strategy, the optimal model can be identified by

$$\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i,f(x_i)) \tag{1.13}$$

where $\mathcal{F}$ is the hypothesis space.
ERM works well when the sample size is large. It has been widely used in many specific situations; maximum likelihood estimation is a good example. However, over-fitting occurs when the sample size is small, which means the model is too tailored to the particularities of the training data and generalisation to new data is poor. Structural risk minimisation (SRM) prevents over-fitting by adding a regulariser or penalty term that indicates the complexity of the model. SRM is denoted by

$$R_{srm}(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \tag{1.14}$$

where $J(f)$ is the complexity of the model and a function of hypothesis space $\mathcal{F}$. $J(f)$ should increase when a model f is more complex and vice versa, which means the degree of complexity of a model determines the penalty term. $\lambda \geq 0$ is a coefficient balancing EMP and the penalty term. A model with the minimum structural risk is usually able to perform well with new data and training data.
Maximum posterior probability (MAP) is a good example of SRM because the model for MAP is a conditional probability distribution; loss function is a log-likelihood loss function and the penalty term is a prior probability.
Thus, the optimal model can be identified by

$$\min_{f} \in \mathcal{F} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f) \tag{1.15}$$

3. Algorithm

According to the standards described in **Strategy**, statistical learning could choose the optimal model from the hypothesis space. The last decision is what kind of algorithm can be used for the solution. Statistical learning is an optimisation problem. The problem would be easy if there were an analytic solution. However, analytic solutions are rare in statistical learning, so numerical analysis is needed to find a solution. How to make sure that the global optimization can be found effectively becomes an important problem.

1.2.4 Evaluation

The most common way to evaluate a classifier is accuracy, which can be defined as: Given a test data set, accuracy is the ratio between the number of samples correctly classified and the number of all samples.
As for binary classification, precision and recall are commonly adopted. These two measures usually consider the class researchers care about as positive class, so there would be four different cases according to classifiers' judgement.

- **True Positive (TP)**: A positive sample is classified as positive.
- **False Negative (FN)**: A positive sample is classified as negative.
- **False Positive (FP)**: A negative sample is classified as positive.
- **True Negative (TN)**: A negative sample is classified as negative.

Precision P can be defined as the follows.

$$P = \frac{TP}{TP+FP} \tag{1.16}$$

Recall R is

$$R = \frac{TP}{TP+FN} \tag{1.17}$$

F-measure is the harmonic mean of precision and recall. Its parameter β is the weight assigned to precision over recall.

$$F_\beta = (1+\beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \tag{1.18}$$

1.3 Topic Models

Topic analysis is currently gaining popularity and reputation in both machine learning and its application [LKE08, HER09, MSH08, LLB08]. A topic model is normally defined as a set of approaches for discovering the latent information in a corpus. Ordinarily, the way to model a document is to map this document to a vocabulary, by creating the matrix of document to word. Each element of the matrix could be weighted by different schemes. The advantage of this approach is to reduce documents of arbitrary length to a fixed-length list of numbers. However, these schemes of modelling documents have the following shortcomings: (i) they can not reveal the inner statistical

structure of a document and (ii) their potential for reduction for dimensionality of the matrix is relatively small. Topic models aim to represent documents in a more abstract way. Imagine the composing process of an article: people can be assumed to have a few topics in their minds first, then to find the words related to the topics as the second step. However, the reader will only see the words on the paper rather than the topics hidden in the author's mind. Topic models attempt to simulate this process and finally reveal these hidden topics and their relations to the words.
To the best of our knowledge, the very first topic model presented in [PTRV98], called Latent Semantic Indexing, makes use of singular value decomposition (SVD) to discover the semantic information in a corpus. Probabilistic Latent Semantic Indexing (pLSI) [Hof99] applies probabilistic techniques to analyse co-occurrence data. The latest refinement is LDA, which is a modification of pLSA.

1.3.1 Mixture Models

In statistics, a mixture model is a probabilistic model for representing the presence of sub-populations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs [Din08]. These sub-populations normally refer to the same type of distribution with different parameters. A multinomial distribution will indicate which sub-population generates a specific sample. Since the relationship between a sample and its sub-population is not observable, these indicators are called latent variables.
A straightforward explanation for a mixture model can be tossing coins. Assume that there are three different coins A, B and C, where the probability of heads p or tails $1-p$ for each coin is unknown. Our rules for tossing the coins are the following:

1. Toss coin A and observe the result.
2. If the result of coin A is head then toss B, else toss C.
3. Write down the result of B or C and repeat this for a finite number of times.

A sequence of observations like "1101001010..." would be obtained after repeating this process. Our aim is to identify parameters p_A, p_B and p_C for this model without knowing everything except for this sequence.
This model can be applied to text modelling as long as the right probability distribution is chosen for modelling the words. Unlike tossing coins, composing a document using

a mixture model is more like a process of rolling dice which have hundreds of sides representing different words and topics. The general idea of this is to replace coins by the die A, though parameter K, the number of topics, has to be specified initially. The process of generating a document is the following.

1. Roll A and observe the result (obtain a topic).
2. Roll the corresponding topic die T_i according to the result of step 1.
3. Write down the words generated by the topic dice.

Similarly, our aim is to estimate the probability for each side of the dice.

1.3.2 Parameters Estimation

In statistics, estimation deals with identifying the value of parameters based on empirical data (observation). For example, an observation of "1110101001" can be generated by tossing a single coin ten times. The probability of heads p can be estimated by using Maximum Likelihood Estimators (MLE), which is an easy and common way to identify parameters without using prior distribution and loss function. MLE chooses as the estimate of p the value of p that provides the largest value of the likelihood function. As for this case, suppose that random variables $X_1, ..., X_n (n = 10)$ form the observation from the Bernoulli distribution with parameter p which is unknown $(0 \leq p \leq 1)$. For all observed values $x_1, ..., x_n$, where each x_i is either 0 or 1, the likelihood function is

$$f_n(\mathbf{x}|p) = p^{x_1}(1-p)^{1-x_2}p^{x_2}(1-p)^{1-x_2}...p^{x_n}(1-p)^{1-x_n} = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} \quad (1.19)$$

In order to maximise the likelihood function, we need now to solve the derivative $\mathrm{d}f_n(\mathbf{x}|p)/\mathrm{d}p$ equal to zero.
Parameter estimation for mixture models is much more difficult, since mixture models contain a layer of latent variables which can not be directly observed. Thus, some iterative methods such as expectation maximisation (EM), Monte Carlo Markov Chain (MCMC) and variational inference have been developed to solve this problem. Although these methods are based on different principles, they have a lot in common. In general, the latent variables will be set to random values initially. Then, an iterative process will be executed until the latent variables finally converge. However, the final results are usually not repeatable due to the randomised initial settings. This problem will be addressed in Section 3.2.2

1.4 Summary

In this chapter, the need for automating the screening phase in systematic reviews was discussed. Potential techniques including machine learning and text mining were introduced. The basic concepts of automatic text classification were presented to show their potential to reduce the burden to reviewers. However, a few drawbacks of the BOW model adopted in text classification need to be addressed. The topic model is believed to solve these problems by generating higher and more abstract features for documents based on their content.
In the next chapter, more detail about the latest classifiers and topic models will be introduced.

1.5 Research Aims

The research presented in this thesis aims to explore how topic models could assist the screening phase powered by automatic text classification in order to construct reviews of better quality. More specifically, the objectives are:

- To investigate whether topic models can be applied to text classification in support of the screening phase in systematic reviews;
- To compare the performance of two methods of text classification: one based on LDA topics and the other employing the BOW model;
- To evaluate the impact of different numbers of topics on topic-based classification.

Chapter 2

Background

Supporting systematic reviews using machine learning and text mining techniques involves two key methods. One is statistical text classification and the other is topic modelling. As mentioned in the Chapter 1, automatic text classification is able to dramatically reduce the burden on reviewers and at the same time maintain high quality of the reviews. However, the BOW model is usually employed as the representation of documents and the basis of text classification, which may be problematic. The latest topic models are an ideal replacement for BOW model since topics are much more informative for both reviewers and machines.
In Chapter 2, the internal mechanism and theory of these methods will be presented.

2.1 Text Classification

All text classification methods aim at predicting the right label for the documents. where the labels are a fixed number of pre-defined classes[Joa98]. Classification methods can be supervised or unsupervised (clustering). This thesis only discusses supervised text classification which requires external feedback or labels to accomplish the training phase of classifiers. In this section, five representative classifiers will be introduced in order to give a comprehensive view of the techniques used in this research.

2.1.1 Perceptron

The perceptron [Ros58] is a linear model for binary classification problems, which takes the feature space of training instances as input and outputs a binary value (class) for each instance, namely +1 or -1. The perceptron separates the instances into positive

and negative by identifying a hyperplane between them. Therefore, a loss function based on error classification should be minimised by using an optimisation algorithm, gradient descent in order to identify the hyperplane. The perceptron can be easily implemented and understood and it is also the basis of neural network and support vector machines (SVM).

Definition

Assuming the input space (feature space) is $\mathcal{X} \subseteq \mathbf{R}^n$, the output space is $\mathcal{Y} = \{+1, -1\}$. $x \in \mathcal{X}$ denotes the feature space of instances; correspondingly, $y \in \mathcal{Y}$ denotes the class of instances. The input space can be mapped to the output space by the following function [Han12]

$$f(x) = sign(w \cdot x + b) \tag{2.1}$$

where w and b are the parameters of the perceptron, $w \in \mathbf{R}^n$ is weight or weight vector and $b \in \mathbf{R}$ is bias. Sign is a signum function which is

$$sign(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases} \tag{2.2}$$

The perceptron has the following geometric meaning: Linear equation 2.3 corresponds to a hyperplane on the feature space $\mathbf{R}^n$, where w and b denote normal vector and intercept of this hyperplane which separate the feature space into two sub-spaces. Points (feature vector) within the two sub spaces, therefore, are classified into positive or negative instances, according to the separating hyperplane, as shown in Figure 2.1.

$$w \cdot x + b = 0 \tag{2.3}$$

In using the training data $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ (feature space and class of instances), the purpose of the training model is to estimate parameters w and b.

Strategy

A necessary assumption that needs to be made is that the training data set are linearly separable, which means, given a data set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, a hyperplane S, $w \cdot x_i + b = 0$ can completely and correctly separate the positive and negative instances of T into the two sides of the hyperplane. To be specific, for each instance i that has $y_i = +1$, $w \cdot x + b > 0$ holds. For each instance i that has $y_i = -1$, $w \cdot x + b < 0$

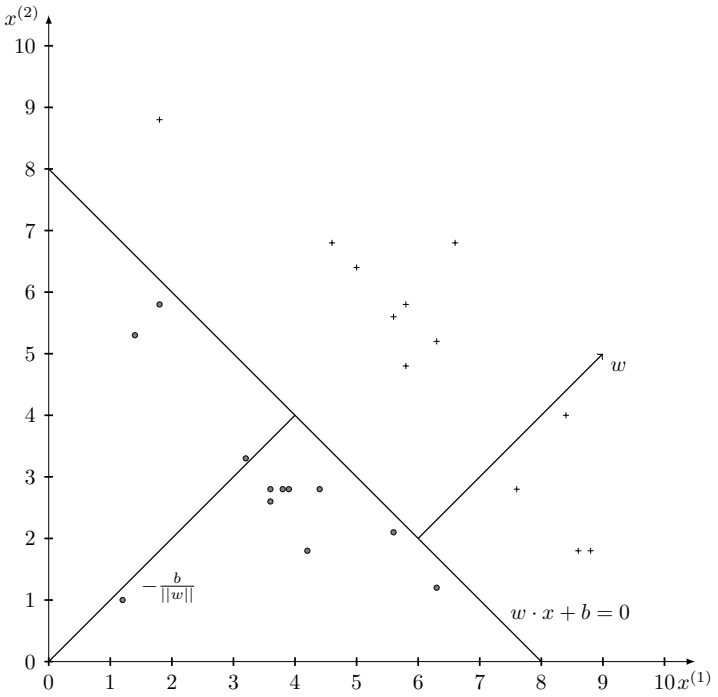

Figure 2.1: Perceptron model

holds.

Assuming that the data set is linearly separable, the hyperplane S is the aim of training the perceptron. Thus, the training strategy is to identify the loss function and minimise it. An easy and natural option for loss function is exploiting the number of instances which are incorrectly classified. However, it is not easy to optimise them since such a loss function can not be derived on w and b. Another option is employing the distance from the instances of misclassification to the hyperplane. Thus, the distance from instance x_0 in input space $\mathbf{R}^n$ to hyperplane S can be expressed by

$$\frac{1}{||w||}|w \cdot x_0 + b| \tag{2.4}$$

where $||w||$ is the L_2 norm of w. As for the instance (x_i, y_i) of misclassification,

$$-y_i(w \cdot x_i + b) > 0 \tag{2.5}$$

holds. Because $y_i = -1$ if $w \cdot x_i + b > 0$, while $y_i = +1$ if $w \cdot x_i + b < 0$. Therefore, the distance from the instance of misclassification x_i to the hyperplane is

$$-\frac{1}{||w||} y_i (w \cdot x_i + b) \tag{2.6}$$

Suppose that set M is a collection of the instances of misclassification in terms of hyperplane S, so the aggregate of distances of all instances misclassified is

$$-\frac{1}{||w||} \sum_{x_i \in M} y_i (w \cdot x_i + b) \tag{2.7}$$

Without $-\frac{1}{||w||}$, the loss function Formula 2.8 can be obtained for teaching the perceptron, given training data set T.

$$L(w,b) = \sum_{x_i \in M} y_i (w \cdot x_i + b) \tag{2.8}$$

Obviously, $L(w,b)$ is a non negative function which equals 0 if there is no misclassification of instances.

Algorithm

The learning problem of the perceptron has now been transferred to an optimisation problem in terms of Function 2.8, which can be solved by using stochastic gradient descent. The problem can be defined by the expression below. Given a training data set

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{2.9}$$

where $x_i \in \mathcal{X} = \mathbf{R}^n$, $y_i \in \mathcal{Y} = \{-1, 1\}$, $i = 1, 2, ..., n$, we aim to estimate the values of w, b, which, in fact, is the solution of the optimal loss Function 2.10

$$\min_{w,b} L(w,b) = \sum_{x_i \in M} y_i (w \cdot x_i + b) \tag{2.10}$$

The process of the perceptron training is driven by misclassification, and uses stochastic gradient descent to minimise target Function 2.10 step by step. The first step is to randomly choose a hyperplane by initialising the values of w and b randomly. The minimisation process is choosing one instance in M one time, which enables gradients of the loss function to drop.

The gradient is a generalisation of the usual concept of derivative of a function in one dimension to a function in several dimensions. The gradients of the loss function are given by $\nabla_w L(w,b)$ and $\nabla_b L(w,b)$.

$$\nabla_w L(w,b) = -\sum_{x_i \in M} y_i x_i \tag{2.11}$$

$$\nabla_b L(w,b) = -\sum_{x_i \in M} y_i \tag{2.12}$$

Randomly choose an instance (x_i, y_i) from misclassification set M and update w and b.

$$w \leftarrow w + \eta y_i x_i \tag{2.13}$$

$$b \leftarrow b + \eta y_i x_i \tag{2.14}$$

where $\eta(0 < \eta \leq 1)$ is step length or learning rate in machine learning, which means how much the hyperplane would adjust itself in every step. During the iteration, the value of loss function $L(w,b)$ will become smaller and smaller until it equals zero. The steps in detail can be given as follows
Input: Training data $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$; Learning rate $\eta(0 < \eta \leq 1)$
Output: w and b; The perceptron $f(x) = sign(w \cdot x + b)$

1. Initialise w_0 and b_0.
2. Choose an instance (x_i, y_i) from the training data set.
3. If $y_i(w \cdot x_i + b) < 0$, update w and b using Formulae 2.13 and 2.14.
4. Start from step 2, until there is no instance that is misclassified.

An intuitive explanation for the perceptron is: the hyperplane would adjust itself more closely to the instance that is misclassified in order to reduce the distance between them. After a limited number of iterations, an appropriate position for the hyperplane will be finally identified if the given training data set is linearly separable. This was proved by Novikoff in 1963 [Nov63].

2.1.2 K-Nearest Neighbour

K-nearest neighbour (K-NN) [Cov68] is a basic classification and regression model, but in this thesis we only discuss the classification model. K-NN takes feature vectors of instances as input, and outputs the classes for instances, which can be considered as

an instance-based learning algorithm. K-NN classifies a candidate instance according to the classes of k nearest instances in training data set around this candidate. Therefore, there is no obvious learning process for K-NN.

Algorithm

The algorithm of K-NN is easy and intuitive. Given training data set T and a candidate instance i, which class i belongs to depends on the majority of classes of instances which are the k nearest neighbour to i.
Input: Training data $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $x_i \in \mathcal{X} \subseteq \mathbf{R}^n$ is feature vector, $y_i \in \mathcal{Y} = \{c_1, c_2, ..., c_K\}$ is class of instance; Feature vector of candidate instance x
Output: Class of instance x.

1. According to a certain type of measure for distance, identify k nearest training instances around the candidate x. The neighbourhood for x that covers these training instances is designated by $N_k(x)$

2. According to the pre-defined classification rule (for example, majority voting rule), decide class y for x:

$$y = arg\max_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), \quad i = 1, 2, ..., N; \quad j = 1, 2, ..., K \tag{2.15}$$

In Formula 2.15, I denotes an indicator function which means if $y_i = c_i$, I is 1, otherwise I is 0. Thus, no training phase is necessary for K-NN since it directly uses the training data as evidence for classification.

Model

After the training data set, distance measurement (for example, the Euclidean distance), the value of k and the classification rule (for example, majority voting rule) are all decided, the class of a new input instance can also be uniquely decided. This process is actually separating the feature space according to the rules mentioned and determines which class instances belong to.
This process can be observed more intuitively when $k = 1$. In feature space, for each instance x_i, the points closer to this instance than others form an area, called cell. Thus, each training instance would own a cell and each cell is a division of the feature space.

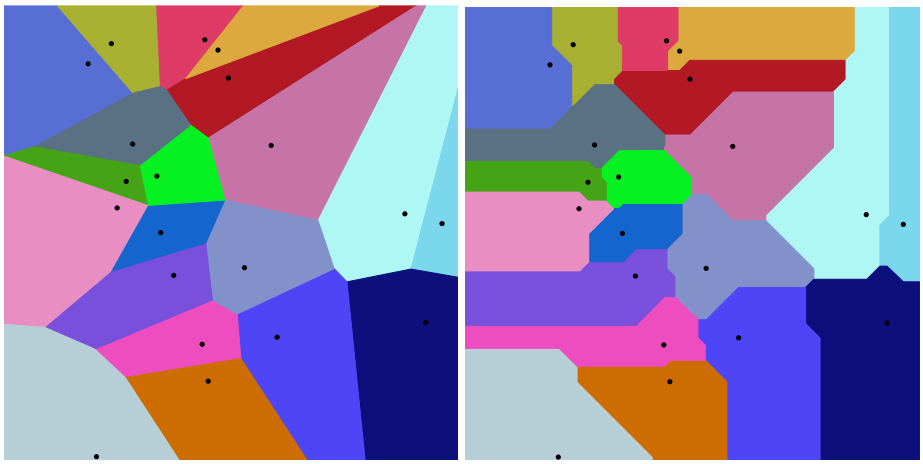

Figure 2.2: Space division for K-NN ($k = 1$) using different distance measurements. **Left**: Euclidean distance. **Right**: Manhattan distance.
Source: en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

When $k = 1$, the classification rule becomes that using class y_i of x_i to label the new instance based in the cell where x_i is. Figure 2.2 shows examples of space division in 2-dimensions.

Distance Measurement

The distance between two instances in feature space reflects their similarity. The feature space for K-NN is normally defined on n-dimensional vector space $\mathbf{R}$, where Euclidean distance is adopted. Besides Euclidean distance, there are other distance measures which can be used for any specific purpose, such as L_p distance or Minkowski distance.
Suppose that feature space $\mathcal{X} \subseteq \mathbf{R}^n$, $x_i, x_j \in \mathcal{X}$, $x_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(n)})$,
$x_j = (x_j^{(1)}, x_j^{(2)}, ..., x_j^{(n)})$, the L_p distance between x_i and x_j is defined as follows:

$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}|^p \right)^{\frac{1}{p}} \tag{2.16}$$

where $p \geq 1$. If $p = 2$, it is called Euclidean distance, which is expressed as

$$L_2(x_i, x_j) = \left(\sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}|^2 \right)^{\frac{1}{2}} \tag{2.17}$$

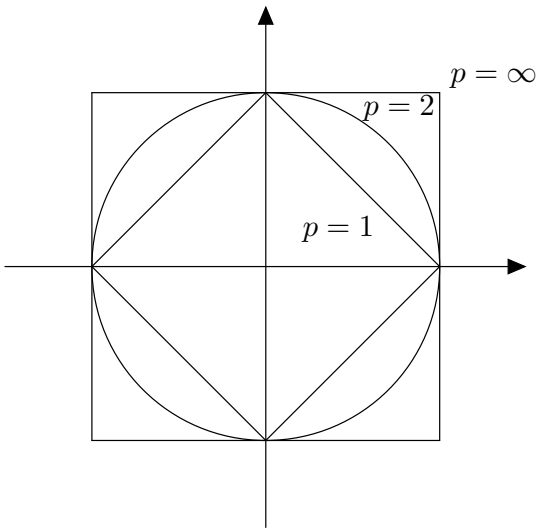

Figure 2.3: Relationship of L_p distances

If $p = 1$, it is called Manhattan distance (See the right of Figure 2.2), which is expressed as

$$L_1(x_i, x_j) = \sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}| \tag{2.18}$$

If $p = \infty$, it is the maximum value of each dimension.

$$L_\infty(x_i, x_j) = \max_l |x_i^{(l)} - x_j^{(l)}| \tag{2.19}$$

Figure 2.3 illustrates that, in 2-dimension space, the L_p distance from the origin point to the points that make $L_p = 1$

Choosing the value of k

Varying the value of k would have a significant impact on the performance of K-NN. If a small value is selected for k, the approximation error is reduced. However, a drawback is that this will make prediction really sensitive to its neighbour instances, which means the results could be easily influenced by noise. More specifically, a small k value means the model becomes more complicated and the over-fitting problem is more likely to occur.

Using a bigger value of k would reduce the estimation error, but it increases the approximation error, which becomes a trade-off problem. Thus, in a particular application, k

usually starts with a small value and then cross-validation is used for choosing the best value for k.

Classification Rule

The majority voting rule is the most common classification rule used in K-NN. The majority of k neighbour instances would determine the class of the candidate instance. Majority voting can be expressed as follows: if the loss function of classification is 0-1 loss function, the classification function is

$$f : \mathbf{R}^n \rightarrow \{c_1, c_2, ..., c_K\} \tag{2.20}$$

Then, the probability of misclassification is

$$P(Y \neq f(X)) = 1 - P(Y = f(X)) \tag{2.21}$$

Given an instance $x \in \mathcal{X}$, $N_k(x)$ denotes k nearest neighbour instances of x. Thus, the rate of misclassification is

$$\frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i \neq c_j) = 1 - \frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i = c_j) \tag{2.22}$$

Our aim is to minimise Formula 2.22 (i.e. empirical risk), in other words, maximise $\frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i = c_j)$. Henceforth, majority voting is the same as the minimisation of empirical risk.

2.1.3 Naive Bayes

The Naive Bayes classifier (NBC) is based on applying Bayes' theorem [DS02] with strong independence assumptions between the features. Given a training data set, NBC first learns the joint probability of input and output, according to independence assumptions. Then, based on the model, NBC would output class y with the maximum posterior using Bayes' theorem. NBC can be easily implemented and maintain high efficiency of learning and prediction so that it has been widely used in machine learning.

Basic Method

Assume that input space $\mathcal{X} \subseteq \mathbf{R}^n$ is a set of n-dimension vector space. The output space is a set of labels $\mathcal{Y} = \{c_1, c_2, ..., c_K\}$. An input for NBC is a feature vector $x \in \mathcal{X}$ and the corresponding output is a label $y \in \mathcal{Y}$. X is a random variable defined at the input space $\mathcal{X}$, similarly, Y is a random variable defined at the output space $\mathcal{X}$. $P(X,Y)$ is the joint probability of X and Y. Thus, the instances in training data set

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{2.23}$$

are i.i.d at $P(X,Y)$.
NBC learns the joint probability distribution $P(X,Y)$, according to the training data. More specifically, it learns it by first looking at the following prior distribution and the conditional distribution. The prior distribution is

$$P(Y = c_k), \quad k = 1, 2, 3, ..., K \tag{2.24}$$

The conditional distribution is

$$P(X = x|Y = c_k) = P(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, ..., X^{(n)} = x^{(n)}|Y = c_k), \quad k = 1, 2, 3, ..., K \tag{2.25}$$

The quantity of parameters for the conditional distribution can be exponential if each instance is sampled from different distributions. This is the reason why the independent assumption is necessary and why this model is naive. Specifically, the assumption can be expressed as follows

$$\begin{aligned} P(X = x|Y = c_k) &= P(X^{(1)} = x^{(1)}, ..., X^{(n)} = x^{(n)}|Y = c_k) \\ &= \prod_{j=1}^{n} P(X^{(j)} = x^{(j)}|Y = c_k) \end{aligned} \tag{2.26}$$

In fact, NBC learns the mechanism of generating samples, which makes it a generative model. The independence assumption means the features that are used for classification are all independent and identically distributed. This would makes NBC easy, but sometimes accuracy is the cost of this advantage.
Given an input x, NBC would output the posterior probability $P(Y = c_k|X = x)$ according to the model that it learned before. Then, the class with the highest posterior probability will be assigned to x as its final prediction. The posterior probability can

be computed according to Bayes' theorem.

$$P(Y = c_k|X = x) = \frac{P(X = x|Y = c_k)P(Y = c_k)}{\sum_k P(X = x|Y = c_k)P(Y = c_k)} \tag{2.27}$$

Inserting Formula 2.34 into Formula 2.27, we have

$$P(Y = c_k|X = x) = \frac{P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)}|Y = c_k)}{\sum_k P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)}|Y = c_k)}, \quad k = 1,2,...,K \tag{2.28}$$

which is the basic formula for NBC. Thus, NBC can be expressed as follows

$$y = f(x) = arg\max_{c_k} \frac{P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)}|Y = c_k)}{\sum_k P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)}|Y = c_k)}, \quad k = 1,2,...,K \tag{2.29}$$

Notice that in Formula 2.29, the denominators for each c_k are the same so that this formula can be simplified as follows

$$y = arg\max_{c_k} P(Y = c_k)\prod_j P(X^{(j)} = x^{(j)}|Y = c_k) \tag{2.30}$$

NBC assigns the class with the highest posterior probability to the candidate instance, which equals the minimisation of expectation risk. Assume the 0-1 loss function is selected,

$$L(Y, f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases} \tag{2.31}$$

where $f(x)$ is the decision function. The expectation risk function is

$$R_{exp}(f) = E[L(Y, f(x))] \tag{2.32}$$

where the expectation is for the joint probability distribution $P(X,Y)$. Thus, we have

$$R_{exp}(f) = E_X \sum_{k=1}^{K} [L(c_k, f(X))]P(c_k|X) \tag{2.33}$$

In order to minimise the expectation risk, all we need to do is minimise $X = x$ one by one, thus, we have

$$\begin{aligned} f(x) &= arg\min_{y\in\mathcal{Y}} \sum_{k=1}^{K} L(c_k, y) P(c_k|X = x) \\ &= arg\min_{y\in\mathcal{Y}} \sum_{k=1}^{K} P(y \neq c_k|X = x) \\ &= arg\min_{y\in\mathcal{Y}} (1 - P(y = c_k|X = x)) \\ &= arg\max_{y\in\mathcal{Y}} P(y = c_k|X = x) \end{aligned} \tag{2.34}$$

By performing the transformation, the minimisation of the expectation risk becomes the maximisation of posterior probability that is adopted by NBC.

$$f(x) = arg\max_{y\in\mathcal{Y}} P(y = c_k|X = x) \tag{2.35}$$

Maximum-likelihood Estimation

In NBC, the process of teaching a model means the estimation of $P(Y = c_k)$ and $P(X^{(j)} = x^{(j)}|Y = c_k)$ can be performed by using Maximum-likelihood Estimation (MLE). The maximum likelihood of prior probability $P(Y = c_k)$ is

$$P(Y = c_k) = \frac{\sum_{i=1}^{N} I(y_i = c_k)}{N} \quad k = 1, 2, ..., K \tag{2.36}$$

Assume the set $a_{j1}, a_{j2}, ..., a_{jS_j}$ is the possible values for the j_{th} feature $x^{(j)}$. The maximum likelihood of the conditional probability $P(X^{(j)} = a_{(jl)}|Y = c_k)$ is

$$\begin{gathered} P(X^{(j)} = a_{(jl)}|Y = c_k) = \frac{\sum_{i=1}^{N} I(x_i^{(j)} = a_{jl}, y_i = c_k)}{\sum_{i=1}^{N} I(y_i = c_k)} \\ j = 1, 2, ..., n; \quad l = 1, 2, ..., S_j; \quad k = 1, 2, ..., K \end{gathered} \tag{2.37}$$

where $x_i^{(j)}$ is the j_{th} feature of the i_{th} instance; a_{jl} is the l_{th} value that can be assigned to j_{th} feature; I is the indicator function.

Algorithm

The learning and classification algorithm of NBC is given below.
Input: Training data $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$.
Output: The class of instance x

1. Compute the prior probability as well as the conditional probability using Formulae 2.36 and 2.37

2. As for the given instance $x = (x^{(1)}, x^{(2)}, ..., x^{(n)})$, compute

$$P(Y = c_k) \prod_{j=1}^{n} P(X^{(j)} = x(j) | Y = c_k), \quad k = 1, 2, ..., K \tag{2.38}$$

3. Output the class of instance x,

$$y = arg \max_{c_k} P(Y = c_k) \prod_{j=1}^{n} P(X^{(j)} = x^{(j)} | Y = c_k) \tag{2.39}$$

2.1.4 Support Vector Machines

Support Vector Machines (SVM) are classifiers that deal with the binary classification problem [CV95]. The basic model of SVM aims to find the maximum-margin hyperplanes, which makes it different from the perceptron. Kernel tricks are also another important aspect for SVM. Kernels enable SVM to deal with non-linear problems. The learning strategy of SVM is maximising the margin of hyperplanes.
The algorithms for developing a SVM can be introduced from the simplest to the most complicated: linear support vector machine in linearly separable case, linear support vector machine and non-linear support vector machine. The simplest one can be considered as the basis of the more complex models. If the training data is linearly separable, hard margin maximisation can be applied. If the training data is nearly linearly separable, soft margin maximisation would be applied. If the training data is not linearly separable, SVM would employ kernel functions and soft margin maximisation.

Support Vector Machine and Hard Margin Maximisation

We consider that this is a binary classification problem and assume the input space and the feature space are two different spaces. Input space can be a Euclidean space or a discrete set while feature space can be a Euclidean space or a Hilbert space. Linear

SVMs in linearly separable cases assume the elements in these two spaces are one-to-one corresponded and map the inputs to the feature space. Non-linear SVMs map these elements from input space to feature space by using a non-linear function.
Given a training data set

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\} \tag{2.40}$$

the linearly separable assumption for the training data set is also required. Our aim is to identify a hyperplane in the feature space, which is able to separate the instance in training data correctly. The hyperplane can be defined as $w \cdot x + b = 0$, where w is a normal vector and b is intercept. Normally, if the training data set is linearly separable, there can be an infinite number of separating hyperplanes. The perceptron utilizes the minimisation of misclassification to identify the hyperplane that has infinite possibilities. However, linear SVMs for linear separable data employ different strategies, maximum margin, which produce a unique separating hyperplane.
Therefore, given a training data set, the model using maximum margin strategy to identify the unique hyperplane $w^\star \cdot x + b^\star = 0$ and the decision function $f(x) = sign(w^\star \cdot x + b^\star)$ is called a linear SVM in linear separable data.
Figure 2.4 demonstrates a binary classification problem in 2-dimension space, where circles stand for positive instances and crosses are negative instances. The training data seems to be linearly separable, while there are many possible lines that can correctly separate the instances. However, SVM would not only identify the separating hyperplane, but it can also find the unique one with the maximum margin.

Margin and Geometry Margin

In Figure 2.4 points A, B and C stand for three instances and they are all positive. Notice that point C is much further away from the separating hyperplane than the points B and A. This is to say, it would have more confidence to predict C as a positive instance, compared to B and A. As B is located between A and C, the confidence of it being a positive instance is bigger than A and smaller than C. Generally, the distance of an instance to the separating hyperplane can be considered as the confidence of the prediction. Assuming the hyperplane $w \cdot x + b = 0$ is known, $|w \cdot x + b|$ is able to relatively represent the distance of x from the hyperplane. Whether $w \cdot x + b$ is positive or negative can be used for indicating if the prediction is correct. Thus, the quantity $y(w \cdot x + b)$ means if a prediction is correct and how correct (confident) it is, which, in

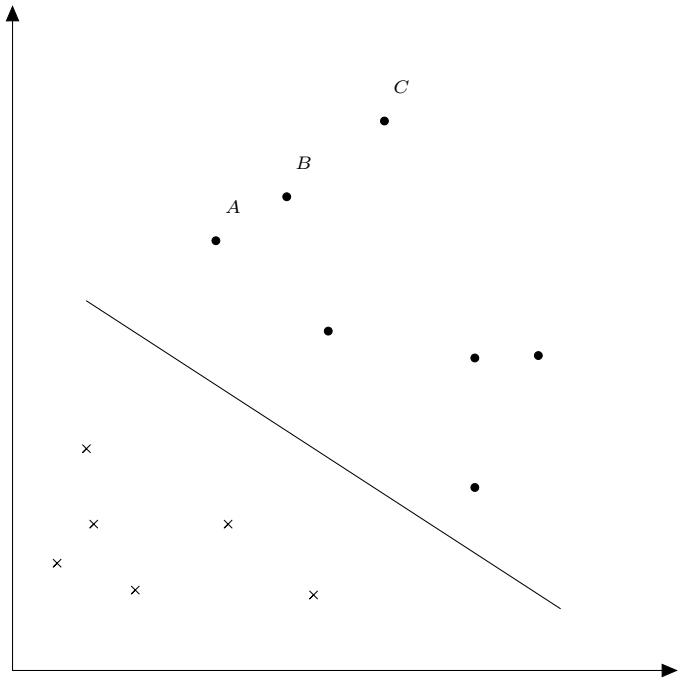

Figure 2.4: Binary classification problem

fact, is the concept of functional margin.
For the training data set T and the separating hyperplane (w,b), the functional margin in terms of (w,b) and instance (x_i,y_i) is defined by

$$\hat{\gamma}_i = y_i(w \cdot x_i + b) \tag{2.41}$$

Also, the functional margin can be defined by the distance of the closest instance in the training data set T to the separating hyperplane (w,b), which is

$$\hat{\gamma} = \min_{i=1,2,\ldots,N} \hat{\gamma}_i \tag{2.42}$$

The functional margin is able to indicate correctness and confidence. In contrast, using only the functional margin is not enough to identify the unique separating hyperplane, because the proportionate changing of w and b, for example, changing w and b to $2w$ and $2b$, would have no impact on the hyperplane, although the functional margin becomes twice as big. This fact tells us that it is necessary to apply a constraint to w such as $||w|| = 1$, which would identify the margin. This makes the functional margin

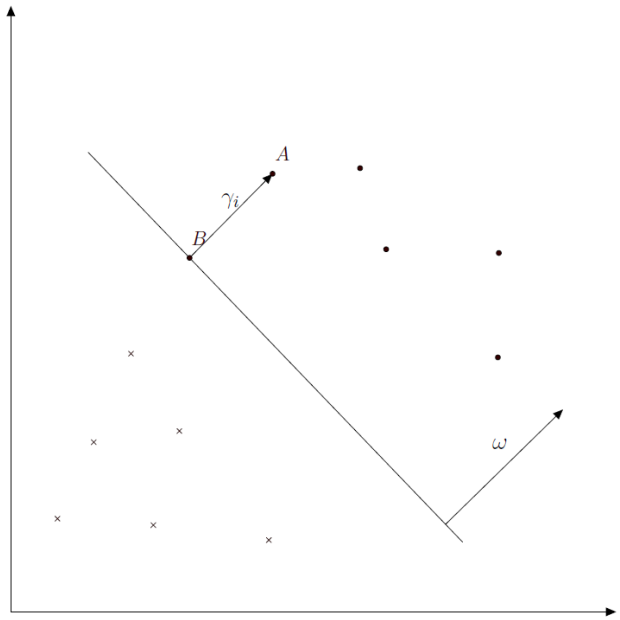

Figure 2.5: Geometric margin

become geometric margin.
Figure 2.5 demonstrates the norm vector w of (w,b). Point A stands for instance x_i and the class is $y_i = +1$. The distance of A to (w,b) is given by segment AB, noted as γ_i.

$$\gamma_i = \frac{w}{||w||} \cdot x_i + \frac{b}{||w||} \tag{2.43}$$

This is the case when point A is located on the positive side of the hyperplane. If A is located at the other side ($y_i = -1$), the distance is

$$\gamma_i = -\left[\frac{w}{||w||} \cdot x_i + \frac{b}{||w||}\right] \tag{2.44}$$

Generally, the distance between an instance and the hyperplane is expressed as follows:

$$\gamma_i = y_i\left[\frac{w}{||w||} \cdot x_i + \frac{b}{||w||}\right] \tag{2.45}$$

Maximising Margin

The basic idea of the SVM is to seek a unique hyperplane that can separate instances correctly and maximise the margin. For a linearly separable data set, there can be an infinite number of separating hyperplanes. However, only one hyperplane correctly separates instances with maximum margin. An intuitive explanation for maximising margin is: the hyperplane with maximum margin usually has high confidence to classify instances. This is to say, the hyperplane not only separates instances, but also the most difficult instances (points close to the hyperplane) are separated with relatively high confidence. Such a hyperplane should have a strong classification performance for the unknown instances.

Now, the method of how to identify a hyperplane with maximum geometric margin will be introduced. More specifically, this problem can be expressed as the following optimisation issue with constraints.

$$\begin{aligned} \max_{w,b} \quad & \gamma \\ s.t. \quad & y_i \left[\frac{w}{||w||} \cdot x_i + \frac{b}{||w||} \right] \geq \gamma, \quad i = 1,2,...,N \end{aligned} \tag{2.46}$$

The mathematical description above can be interpreted as: we are going to maximise γ what is the geometric margin in terms of hyperplane (w,b). The constraint is that the geometric margin of hyperplane (w,b) in terms of each training instance should at least equal γ. The problem can be modified as follows, according to the relationship between the geometric and functional margins.

$$\begin{aligned} \max_{w,b} \quad & \frac{\hat{\gamma}}{||w||}\gamma \\ s.t. \quad & y_i(w \cdot x_i + b) \geq \hat{\gamma}, \quad i = 1,2,...,N \end{aligned} \tag{2.47}$$

The value of $\hat{\gamma}$ does not really affect the solution of the optimisation. In fact, if we change w and b proportionally as λw and λb, the functional margin would also become $\lambda\hat{\gamma}$ so that we can set $\hat{\gamma} = 1$. Inserting $\hat{\gamma} = 1$ to the optimisation problem above, we have

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 \\ s.t. \quad & y_i(w \cdot x_i + b) - 1 \geq 0, \quad i = 1,2,...,N \end{aligned} \tag{2.48}$$

which is a convex quadratic programming problem. Further explanation of how to solve this problem and the proof of uniqueness and existence of hyperplanes will not be presented here. Generally, a convex quadratic programming problem can be changed into a dual problem according to Lagrange duality, then, finding the solution using the dual algorithm.
The solution $w^\star$, $b^\star$ allows us to identify the separating hyperplane $w^\star \cdot x + b^\star = 0$ and the decision function $f(x) = sign(w^\star \cdot x + b^\star)$, which is the linear SVM in the linearly separable case.

Algorithm

Input: The linearly separable training data set $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$
Output: The separating hyperplane with maximum margin and the decision function.

1. Build the constrained optimisation problem 2.48 and find the optimal solution $w^\star$, $b^\star$.

2. Obtain the separating hyperplane $w^\star \cdot x + b^\star = 0$ and decision function $f(x) = sign(w^\star \cdot x + b^\star)$.

Linear Support Vector Machines and Soft Margin Maximisation

The learning strategy for linear problems is not applicable to a data set which is not linearly separable. The methods to deal with the non-linear problem include kernel tricks and soft margin maximisation. The latter will be introduced in this section, while the kernel tricks will be discussed in the next section.
Assuming there is a given training data set

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\} \tag{2.49}$$

where $x_i \in \mathcal{X} = \mathbf{R}^n$, $y_i \in \mathcal{Y} = \{+1, -1\}$, $i = 1, 2, ..., N$, x_i stands for the ith feature vector, y_i stands for the class of x_i. Another assumption is that our training data set T is not linearly separable any more, containing some noisy points. This is to say, these noisy points (x_i, y_i) can not allow the function margin to equal or be greater than 1. In order to solve this issue, a slack variable $\xi_i \geq 0$ is introduced for each instance so that the function margin can be greater than 1 after adding a slack variable. This makes the

constraint as follows:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \tag{2.50}$$

Meanwhile, the original target function $\frac{1}{2}||w||^2$ becomes

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \tag{2.51}$$

where $C > 0$ is a penalty function determined by particular applications. Formula 2.51 includes two different meanings: (1) Minimise $\frac{1}{2}||w||^2$ as much as possible (maximising margin), (2) meanwhile, keep the instances that could be misclassified as few as possible. C is a coefficient.
So the learning process for the data set that is not linearly separable becomes a convex quadratic programming problem again.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i \\ s.t. \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1,2,...,N \\ & \xi_i \geq 0, \quad i = 1,2,...,N \end{aligned} \tag{2.52}$$

By transferring 2.52 to a Lagrangian dual problem, the hyperplane $w^\star \cdot x + b^\star = 0$ and the decision function $f(x) = sign(w^\star \cdot x + b^\star)$ can be identified.

2.1.5 Non-linear Support Vector Machine

Linear classifiers are quite effective for linear classification problems. However, non-linear problems may need more complicated models which mainly apply kernel tricks. Non-linear problems can be defined such that the instances will be correctly classified by using a non-linear model. As shown in Figure 2.6, "$\cdot$" represents the positive instance and "$\times$" represents the negative. It is obvious that there is no such line which is able to correctly separate the positive and negative instances. However, they can be separated by an ellipse (non-linear model).
Generally, given a training dataset $T = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where $x_i \in \mathcal{X} = \mathbf{R}^n$, the corresponding label has two classes $y_i \in \mathcal{Y} = \mathbf{R}^n, i = 1,2,...,N$. If there is a hypersurface which is able to separate the instances correctly, this kind of problem can be defined as a non-linearly separable problem.

It is quite difficult to solve a non-linear problem. However, we can transform a non-linear problem into a linear problem by applying a non-linear transformation so that solving the linear problem is equivalent to solving the original non-linear problem. Assuming that the original space is $\mathcal{X} \subset \mathbf{R}^2, x = (x^{(1)}, x^{(2)}) \in \mathcal{X}$, the new space is $\mathcal{Z} \subset \mathbf{R}^2, z = (z^{(1)}, z^{(2)}) \in \mathcal{Z}$. Define a mapping (transformation) from the original space to the new space as follows:

$$z = \phi(x) = ((x^{(1)})^2, (x^{(2)})^2)^T \tag{2.53}$$

The original space $\mathcal{X} \subset \mathbf{R}^2$ is transferred to the new space $\mathcal{Z} \subset \mathbf{R}^2$ as well as the instances and the ellipse using the mapping (2.53). So the ellipse in the original space

$$w_1(x^{(1)})^2 + w_2(x^{(2)})^2 + b = 0 \tag{2.54}$$

becomes a line in the new space:

$$w_1 z^{(1)} + w_2 z^{(2)} + b = 0 \tag{2.55}$$

In the new space, line $w_1 z^{(1)} + w_2 z^{(2)} + b = 0$ is capable of separating the instances correctly, which makes a non-linear problem become a linear one.

According to the description above, the kernel trick follows the same principle and contains two general steps: firstly, a mapping needs to be identified in order to transfer the data from the original space to the new space; secondly, the classifier can be taught in the new space using the learning strategy for the linear model.

When kernel tricks are applied to SVMs, the basic idea is to transfer an input space (Euclidean space $\mathbf{R}^n$) to a corresponding feature space (Hilbert space $\mathcal{H}$) so that a hypersurface in the input space $\mathbf{R}^n$ will correspond to a hyperplane in the feature space $\mathcal{H}$. Therefore, the final model can be trained by solving the linear support vector machine in the feature space.

Kernel Functions

Definition of Kernel Function

Assume that $\mathcal{X}$ is the input space and $\mathcal{H}$ is the feature space. Suppose there exists a mapping from $\mathcal{X}$ to $\mathcal{H}$

$$\phi(x) : \mathcal{X} \longrightarrow \mathcal{H} \tag{2.56}$$

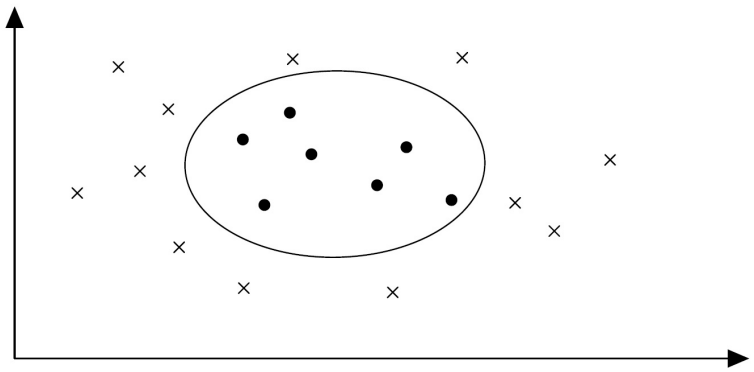

Figure 2.6: Non-linear problems

such that for all $x, z \in \mathcal{X}$, function $K(x,z)$ satisfies

$$K(x,z) = \phi(x) \cdot \phi(z) \tag{2.57}$$

$K(x,z)$ is called the kernel function, where $\phi(x)$ is the mapping function and $\phi(x) \cdot \phi(z)$ is the inner product of $\phi(x)$ and $\phi(z)$. The idea of kernel tricks is that, during the learning process, only $K(x,z)$ will be defined instead of ϕ since $K(x,z)$ is much easier to compute than using $\phi(x)$ and $\phi(z)$.

A linear SVM is usually solved by solving its dual problem. Recalling Formula 2.52 which is the primal problem, the dual problem can be given as follows according to Lagrange duality.

$$\begin{aligned} \min_{a} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i}^{N}\alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C, \quad i = 1,2,...,N \end{aligned} \tag{2.58}$$

It can be noticed that 2.58 only involves the inner product of the instances so that the inner production $x_i \cdot x_j$ in 2.58 can be replaced by $K(x_i,x_j) = \phi(x_i) \cdot \phi(x_j)$.

$$W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j) - \sum_{i=1}^{N}\alpha_i \tag{2.59}$$

This equals to transferring the inputs to a new feature space using ϕ. More specifically, it replaces $x_i \cdot x_j$ in the input space with $\phi(x_i) \cdot \phi(x_j)$. This guarantees that a non-linear classification model will be obtained, as long as the mapping function ϕ is non-linear.

Common Kernel Functions

The kernel functions used in this study will be given as follows:

- Polynomial kernel function

$$K(x,z) = (x \cdot z + 1)^p \tag{2.60}$$

  Decision function

$$f(x) = sign\left[\sum_{i=1}^{N_s} \alpha_i^\star y_i (x_i \cdot x + 1)^p + b^\star\right] \tag{2.61}$$

- Gaussian kernel function (Radial basis function)

$$K(x,z) = exp\{-\frac{\|x-z\|^2}{2\sigma^2}\} \tag{2.62}$$

  Decision function

$$f(x) = sign\left[\sum_{i=1}^{N_s} a_i^\star y_i exp\left[-\frac{\|x-z\|^2}{2\sigma^2}\right] + b^\star\right] \tag{2.63}$$

Algorithm

Input: The training data $T = \{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$, where $x_i \in \mathcal{X} = \mathbf{R}^n$, $y_i \in \mathcal{Y} = -1,+1, i = 1,2,...,N$.
Output:The decision function.

1. Choose the appropriate kernel function $K(x,z)$ and C, create the optimisation

problem.

$$\begin{aligned} \min_{a} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i \\ s.t. \quad & \sum_{i}^{N}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1,2,...,N \end{aligned} \tag{2.64}$$

Find the optimal solution $\alpha^\star = (\alpha_1^\star, \alpha_2^\star, ..., \alpha_n^\star)$

2. Calculate $b^\star = y_j - \sum_{i=1}^{N}\alpha_i^\star y_i K(x_i \cdot x_j)$

3. Build the decision function:

$$f(x) = sign\left[\sum_{i=1}^{N}\alpha_i^\star y_i K(x \cdot x_i) + b^\star\right] \tag{2.65}$$

2.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) was proposed in 2003 [BNJ03], which was the first topic model presented in the form of graphic model. LDA considers that a document or article contains a mixture of various topics. This is similar to probabilistic latent semantic analysis (pLSA). However, a Dirichlet prior has been used in LDA for topics which produces a more reasonable mixture of topics in a document. A typical way to develop a LDA model for a collection of documents is to, firstly, set the number of topics K. Secondly, as LDA is an unsupervised model, it only requires a collection of documents without any external labels. By using a complicated process of estimation, the topic proportion of each document and the topics will be finally presented. Figure 2.7 demonstrates how a topic model works for a document. LDA would discover the inner relationship among words and summarise the documents using the relationship. A topic is actually a cluster of ranked words, where the rank represents the correlation between a word and a topic. So it is quite common to see words that have similar meaning appearing in the same topic (See Figure 2.7).

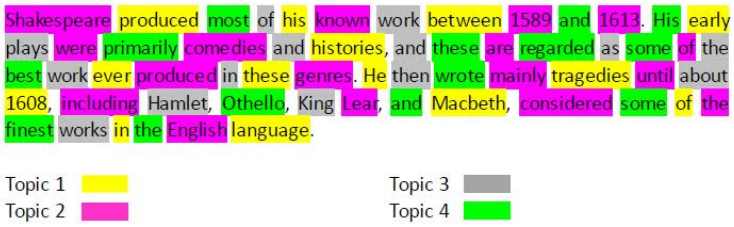

Figure 2.7: Topic model

2.2.1 Generative Process

LDA as a generative model assumes that a document covers a number of topics, and each word in a document is sampled from the probability distributions with different parameters, so each word would be generated with a latent variable to indicate the distribution it comes from. By computing the extent to which each topic is represented in a document, the content of the document can be represented at a higher level, i.e. as a set of topics. The process can be defined by a graphical model (Figure 2.8). Thus, the following steps of generating a document $\vec{w}$ in a corpus D can be given, according to Figure 2.8, while Table 2.2.1 lists all the involved notation.

- Choose K topics $\phi \sim Dir(\vec{\beta})$
- Choose topics proportion $\vec{\theta}_m \sim Dir(\vec{\alpha})$
- Choose a document length $N_m \sim Poisson(\xi)$
- For each word w_n in document m:
    1. Choose a topic $z_{n,m} \sim Multinomial(\vec{\theta}_m)$
    2. Choose a word $w_{n,m}$ from $p(w_{n,m}|\vec{\phi}_{z_{n,m}}, \vec{\theta}_m)$, a multinomial probability conditioned on the topic z_n.

The hyperparameters $\vec{\alpha}$ and $\vec{\beta}$ are the parameters of the prior probability distributions which facilitate calculation. The hyperparameters can be initialised as constant values. They may also be considered as hidden variables which require estimation. The joint

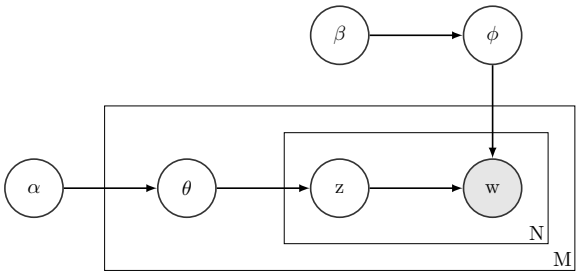

Figure 2.8: Graphic Model: Latent Dirichlet Allocation

probability, i.e. the complete-data likelihood of a document, can be specified [Hei05].

$$p(\vec{w}_m, \vec{z}_m, \vec{\theta}_m, \phi; \vec{\alpha}, \vec{\beta}) = \overbrace{\underbrace{\prod_{n=1}^{N_m} p(w_{n,m}|\vec{\phi}_{z_{n,m}}) p(z_{n,m}|\vec{\theta})}_{\text{words in document}} \cdot p(\vec{\theta}_m|\vec{\alpha})}^{\text{one document}} \cdot \underbrace{p(\phi|\vec{\beta})}_{\text{topics}} \tag{2.66}$$

Table 2.1: Notation in LDA

K	number of topics
$\vec{\alpha}$	hyperparameter on document-topic distribution
$\vec{\beta}$	hyperparameter on topics-word distribution
$\vec{\theta}_m$	a set of parameter vectors for generating a specific topic z in document m
ϕ	a set of parameter vectors for generating word w, according to z
$w_{n,m}$	n^{th} word in document m
$z_{n,m}$	topic indicator for n^{th} word in document m

This is the basis of many other derivations. So the likelihood of a document $\vec{w}_m$ is obtained, which is one of its marginal distributions by integrating $\vec{\theta}_m$ and ϕ out and summing over $z_{m,n}$

$$p(\vec{w}_m|\vec{\alpha}, \vec{\beta}) = \iint p(\vec{\theta}_m|\vec{\alpha}) p(\phi|\vec{\beta}) \cdot \prod_{n=1}^{N_m} \sum_{z_{m,n}} p(w_{m,n}|\vec{\phi}_{z_{n,m}}) p(z_{m,n}|\vec{\theta}_m) \, \mathrm{d}\phi \, \mathrm{d}\vec{\theta}_m \tag{2.67}$$

$$= \iint p(\vec{\theta}_m|\vec{\alpha}) p(\phi|\vec{\beta}) \cdot \prod_{n=1}^{N_m} p(w_{m,n}|\vec{\phi}_{z_{n,m}}) \, \mathrm{d}\phi \, \mathrm{d}\vec{\theta}_m \tag{2.68}$$

Finally, the probability of a corpus is the production of the likelihoods of the independent documents.

$$p(D|\vec{\alpha},\vec{\beta}) = \prod_{m=1}^{M} p(\vec{w}_m|\vec{\alpha},\vec{\beta}) \tag{2.69}$$

2.2.2 Parameters Estimation

The exact inference for LDA is usually intractable. However, this issue can be solved by using approxiamte inference algorithms such as mean-field variational expectation maximisation [BNJ03], expectation propagation [ML02] and Gibbs sampling [Gri02, GS04, PSD00]. Here we introduce Gibbs sampling which is a special case of Markov-chain Monte Carlo (MCMC) [KL51] simulation, because it is a relatively simple algorithm for approximate inference in high-dimensional models such as LDA.
Gibbs sampling can be applied when there are at least two dimensions of dataset X, i.e. each point x is really $x = [x_1, x_2, ..., x_k]$, with $k > 1$. A Gibbs sampler is able to generate samples for those unobservable variables, where each sample depends on the other $k-1$ dimensions. By repeating the sampling process T times, the accuracy of samples increases step by step and finally converges to stable state. The general process of a Gibbs simpler [Hei05] works as follows:

- Randomise values for $x^{(0)} = [x_1^{(0)}, x_2^{(0)}, ..., x_k^{(0)}]$ as the initial state.
- For $t = 1...T$
  - For $i = 1...k$, samping $x_i^{(t+1)}$ according to the following distribution:

$$P(x_i|z_1^{(t+1)}, ...z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_k^{(t)}) = \frac{P(z_1^{(t+1)}, ...z_{i-1}^{(t+1)}, z_i^{(t)}, z_{i+1}^{(t)}, ..., z_k^{(t)})}{P(z_1^{(t+1)}, ...z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_k^{(t)})} \tag{2.70}$$

Using Equation 2.70, the new value for every variable is sampled according to its distribution based on all the other variables. Take an example of three variables:

- The new value for x_1 is sampled conditional on the old values of x_2 and x_3
- The new value for x_2 is sampled conditional on the new value of x_1 and the old value of x_3
- The new value for x_3 is sampled conditional on the new values of x_1 and x_2

The univariate conditionals or full conditionals $p(x_i|\vec{x}_{\neg i})$ have to be found in order to build a Gibbs sampler, using:

$$p(x_i|\vec{x}_{\neg i}) = \frac{p(x)}{p(\vec{x}_{\neg i})} = \frac{p(\vec{x})}{\int p(\vec{x})dx_i} \tag{2.71}$$

where $\vec{x} = [x_i, \vec{x}_{\neg i}]$ and $\vec{x}_{\neg i} = [x_1, x_2, ..., x_{i-1}, x_{i+1}, x_{i+2}, ..., x_n]$. For models like LDA that contain hidden variables $\vec{z}$, the posterior is the target distribution given evidence $p(\vec{z}|\vec{x})$. With Eq. 2.71, the general formulation of a Gibbs sampler for such latent-variable models is given as follows

$$p(z_i|\vec{z}_{\neg i}, \vec{x}) = \frac{p(\vec{z}, \vec{x})}{p(\vec{z}_{\neg i}|\vec{x})} = \frac{p(\vec{z}, \vec{x})}{\int_Z p(\vec{z}, \vec{x})\, \mathrm{d}z_i} \tag{2.72}$$

where the integral changes to a sum for discrete variables.
Building a Gibbs sampler for LDA follows similar methods to those. The hidden variables in LDA are $z_{m,n}$, which is the topic indicator for word $w_{m,n}$. The parameter sets ϕ and θ can be integrated out, since the conjugate distinctions are used in LDA. This technique is called "collapsed" [Nea00],which dramatically facilitates the computing of Gibbs sampler. The target of inference is the distribution $p(\vec{z}|\vec{w})$, which is proportional to the joint distribution

$$p(\vec{z}|\vec{w}) = \frac{p(\vec{z}, \vec{w})}{p(\vec{w})} = \frac{\prod_{i=1}^{W} p(z_i, w_i)}{\prod_{i=1}^{W} \sum_{k=1}^{K} p(z_i = k, w_i)} \tag{2.73}$$

The hyperparameters $\vec{\alpha}$ and $\vec{\beta}$ are omitted here. Of course, they can still be considered as hidden variables for the other applications. Notice that the denominator Eq. 2.73 is hard to evaluate, where the Gibbs sampler would play a significant role. In this case, the desired Gibbs sampler performs a Markov chain that uses the full conditional $p(z_i|\vec{z}_{\neg i}, \vec{w})$ in order to simulate $p(\vec{z}|\vec{w})$. In order to obtain full conditional $p(z_i|\vec{z}_{\neg i}, \vec{w})$, we need to factorise the following joint distribution in LDA.

$$p(\vec{w}, \vec{z}|\vec{\alpha}, \vec{\beta}) = p(\vec{w}|\vec{z}, \vec{\beta}) p(\vec{z}|\vec{\alpha}) \tag{2.74}$$

As the first and second terms are independent of each other, both elements of the joint distribution can be handled separately. By utilising K-dimension Dirichlet-multinomial

conjugation, the target distribution $p(\vec{w}|\vec{z},\vec{\beta})$ can be simply obtained

$$p(\vec{w}|\vec{z},\vec{\beta}) = \prod_{z=1}^{K} \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{\beta})}, \quad \vec{n}_z = [n_z^{(t)}]_{t=1}^{V} \tag{2.75}$$

where the notation $n_z^{(t)}$ denotes the number of times that term t has been observed with topic z.
A similar principle can be applied to the topic distribution $p(\vec{z}|\vec{\alpha})$, so we have

$$p(\vec{z}|\vec{\alpha}) = \prod_{m=1}^{M} \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{\alpha})}, \quad \vec{n}_m = [n_m^{(k)}]_{k=1}^{K} \tag{2.76}$$

The joint distribution therefore becomes:

$$p(\vec{z},\vec{w}|\vec{\alpha},\vec{\beta}) = \prod_{z=1}^{K} \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{\beta})} \cdot \prod_{m=1}^{M} \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{\alpha})} \tag{2.77}$$

The full conditional distribution, i.e. the updating equation from which the Gibbs sampler generates samples, can now be derived from the joint distribution 2.77 with index $i = (m,n)$.

$$\begin{aligned}
p(z_i = k|\vec{z}_{\neg i},\vec{w}) &= \frac{p(\vec{w},\vec{z})}{p(\vec{w},\vec{z}_{\neg i})} && (2.78)\\
&= \frac{p(\vec{w}|\vec{z})}{p(\vec{w}_{\neg i}|\vec{z}_{\neg i})p(w_i)} \cdot \frac{p(\vec{z})}{p(\vec{z}_{\neg i})} && (2.79)\\
&\propto \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{n}_{z,\neg i} + \vec{\beta})} \cdot \frac{\Delta(\vec{n}_m + \vec{\alpha})}{\Delta(\vec{n}_{m,\neg i} + \vec{\alpha})} && (2.80)\\
&\propto \frac{\Gamma(n_k^{(t)} + \beta_t)\Gamma(\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t)}{\Gamma(n_{k,\neg i}^{(t)} + \beta_t)\Gamma(\sum_{t=1}^{V} n_k^{(t)} + \beta_t)} \cdot && (2.81)\\
&\quad \frac{\Gamma(n_m^{(k)} + \alpha_k)\Gamma(\sum_{k=1}^{K} n_{m,\neg i}^{(k)} + \alpha_k)}{\Gamma(n_{m,\neg i}^{(k)} + \alpha_k)\Gamma(\sum_{k=1}^{K} n_m^{(k)} + \alpha_k)} && (2.82)\\
&\propto \frac{n_{k,\neg i}^{(t)} + \beta_t}{\sum_{t=1}^{V} n_{k,\neg i}^{(t)} + \beta_t} \cdot \frac{n_{m,\neg i}^{(k)} + \alpha_k}{\sum_{k=1}^{K} n_m^{(k)} + \alpha_k} && (2.83)
\end{aligned}$$

where the count $n^{(\cdot)}_{\cdot,\neg i}$ denotes that the word i is excluded from the corresponding document m or topic k and the property of Gamma function [1] is used in transformation [2] from Eq. 2.82 to Eq. 2.83.

Finally, applying Bayes's law to the distributions $p(\vec{w}|\vec{z},\phi)$ and $p(\vec{z}|\theta)$, the multinomial parameter sets θ and ϕ can be obtained according to their definitions as multinomial distribution with Dirichlet prior.

$$p(\vec{\theta}_m|\vec{w},\vec{z},\vec{\alpha}) = \frac{1}{Z_{\theta_m}}\prod_{n=1}^{N_m} p(z_{m,n}|\vec{\theta}_m)p(\vec{\theta}_m|\vec{\alpha}) = Dir(\vec{\theta}_m|\vec{n}_m+\vec{\alpha}) \tag{2.84}$$

$$p(\vec{\phi}_k|\vec{w},\vec{z},\vec{\beta}) = \frac{1}{Z_{\phi_k}}\prod_{i:z_i=k} p(w_i|\vec{\phi}_k)p(\vec{\phi}_k|\vec{\beta}) = Dir(\vec{\phi}_k|\vec{n}_k+\vec{\beta}) \tag{2.85}$$

where $\vec{n}_m = \langle n_m^{(1)}, n_m^{(2)}, ..., n_m^{(K)}\rangle$ and $\vec{n}_k = \langle n_k^{(1)}, n_k^{(2)}, ..., n_k^{(V)}\rangle$. Using the exception of the Dirichlet distribution, $\langle Dir(\vec{\alpha})\rangle = \frac{a_i}{\sum_i a_i}$ in Eq. 2.84 and Eq. 2.85 yields:

$$\phi_{k,t} = \frac{n_k^{(t)}+\beta_t}{\sum_{t=1}^{V} n_k^{(t)}+\beta_t} \tag{2.86}$$

$$\theta_{m,k} = \frac{n_m^{(k)}+\alpha_k}{\sum_{k=1}^{K} n_m^{(k)}+\alpha_k} \tag{2.87}$$

2.2.3 Algorithm

So far, three most important equations (Eq. 2.83, Eq. 2.86 and Eq. 2.87) have been obtained for the final algorithm of the Gibbs sampler. The procedure itself only processes five data structures. The first and second ones are count variables $n_m^{(z)}$ and $n_z^{(t)}$ which can be considered as the elements of two matrices. One has $M \times K$ dimensions and the other has $K \times V$ dimensions. The third and fourth ones are their row sums n_m and n_z with dimensions M and K respectively. The last one is the state variable $z_{m,n}$ with dimension W. Here the pseudo codes for the Gibbs sampler are given below:

Input: T The number of iterations; K The number of topics; The document collection D.

Output: Document-topic co-occurrence matrix ϕ and Topic-word co-occurrence matrix θ.

[1] $\Gamma(x+1) = x\Gamma(x)$

[2] The hyperparameters $\vec{\alpha}$ and $\vec{\beta}$ are also omitted here.

1. Zero all count variables, $n_m^{(k)}$, n_m, $n_k^{(t)}$ and n_k.

2. **For** each document m **in** corpus D:

   (a) **For** each word **in** m:

   - Assign a random topic to word $z_{m,n} = k\ Mult(1/K)$.
   - $n_m^{(k)} = n_m^{(k)} + 1$.
   - $n_m = n_m + 1$.
   - $n_k^{(t)} = n_k^{(t)} + 1$.
   - $n_k = n_k + 1$.

3. **For** $i = 0$ **to** T:

   (a) **For** each document m **in** corpus D:

   i. **For** each word t **in** m:

   - For the current assignment k to a word t. Perform $n_m^{(k)} = n_m^{(k)} - 1$; $n_m = n_m - 1$; $n_k^{(t)} = n_k^{(t)} - 1$; $n_k = n_k - 1$.
   - sampling $\vec{p}$ using Eq. 2.83. Generate new topic $\widehat{k}$ for current word t using parameter set $\vec{p}$.
   - Perform $n_m^{(\widehat{k})} = n_m^{(\widehat{k})} + 1$; $n_m = n_m + 1$; $n_{\widehat{k}}^{(t)} = n_{\widehat{k}}^{(t)} + 1$; $n_{\widehat{k}} = n_{\widehat{k}} + 1$.

4. Computing parameter set ϕ and θ, according to Eq. 2.86 and Eq. 2.87.

2.3 Related Works

Besides LDA, there are many other approaches for discovering abstract information from a corpus. Latent Semantic Analysis [BDO95] makes use of singular value decomposition (SVD) to discover the semantic information in a corpus, SVD is a factorisation of matrix which has many applications in statistics and signal processing. Unlike other topic models producing results, an approach [LM14] based on the anchor-word algorithm [AGM12] provides an efficient and visual way for topic discovery. This method first reduces the dimensions of words co-occurrence matrix to 2 or 3, then identifies the convex hull of these words, which can be considered as a rubber band holding these words. The words at anchor points are considered as topics.
Automatic text classification for systematic reviews has been investigated by Bekhuis et al. [BDF10] who focused on using supervised machine learning to assist with the

screening phase. Octaviano et al. [OFMF15] combined two different features, i.e. content and citation relationship between the studies, to automate the selection phase as much as possible. Their strategy reduced workload by 58.2%. Cohen et al. [Coh08] compared different feature representations for supervised classifiers. They concluded that the best feature set used a combination of n-grams and MeSH (Medical Subject Headings)[3] features. Felizardo et al. developed a visual text mining tool that integrated many text mining functions for systematic reviews and evaluated the tool with 15 graduate students [FSM13]. The results showed that the use of the tool is promising in terms of screening burden reduction. Fiszman et al. [FBS+10] combined symbolic semantic processing with statistical methods for selecting both relevant and high quality citations. Frimza et al. [FIM10] introduced a per-question classification method that uses an ensemble of classifiers that exploit the particular protocol used in creating the systematic review. Jonnalagadda et al. [JP13] described a semi-automatic system that requires human intervention. They successfully reduced the number of articles that needed to be reviewed by 6% to 30% while maintaining a recall performance of 95%. Matwin et al. [MKI+10] exploited a factorised complement naive Bayes classifier for reducing the workload of experts reviewing journal articles for building systematic reviews of drug class efficacy. The minimum and maximum workload reduction was 8.5% and 62.2% respectively, and the average over 15 topics was 33.5%. Wallace et al. [WTL+10] showed that active learning has the potential to reduce the workload of the screening phase by 50% on average. Cohen et al. [CWPY06] constructed a voting perceptron-based automated citation classification system which is able to reduce the number of articles that need to be reviewed by more than 50%. Bekhuis et al. [BDF12] investigated the performance of different classifiers and feature sets in terms of their ability to reduce workload. The reduction was 46% for SVMs and 35% for complement naive Bayes classifiers with bag of words extracted from full citations.

From a topic modelling perspective, Miwa et al. [MTOEA14] firstly used LDA to reduce the burden of screening for systematic reviews using an active learning strategy. The strategy utilised the topics as another feature representation of documents when no manually assigned information such as MeSH terms is available. Moreover, Miwa et al. used topic-features for training ensemble classifiers. Similarly, Bekhuis et al. [BTMDF14] investigated how the different feature selections, including topic features, affect the performance of classification.

[3] https://www.nlm.nih.gov/pubs/factsheets/mesh.html

Chapter 3

Topic Modelling for the Screening Phase of Systematic Reviews

In the previous chapters, the basics of text classification were introduced with emphasis on statistical learning. The concept of mixture models was summarised and one of the most well-known models, Latent Dirichlet Allocation, was discussed in detail. In this chapter, we generally consider the screening phase in systematic reviews as a classification problem. We compared two different approaches, the BOW-based approach and the topic-based approach, based on the problem. The BOW-based approach acts as a baseline where the classification is performed using the BOW model. The topic-based approach classifies documents using the topic distribution of each document. Another important aspect is that systematic reviews are mainly applied to biomedical documents which contain many technical terms. In order to make the topics more informative and intuitive, an automatic term recognition (ATR) tool was used to identify these terms in the corpus. The details of these approaches will be presented in the following sections.

3.1 Experimental Design

The aim of the study is to explore the use of topic modelling methods to derive a more informative representation of study, and to determine if this approach works better than traditional BOW representation. The study will be divided into two parts in order to carry out a systematic comparison of the two different approaches to text classification.

$$\langle line \rangle \langle target \rangle \langle feature \rangle : \langle value \rangle ... \langle feature \rangle : \langle value \rangle \# \langle info \rangle$$

Figure 3.1: SVMlight Format

- **BOW-based Approach** The classifiers are trained on the traditional BOW features. The classifier producing the best performance and the best documents encoding method was found in order to act as a baseline.

- **Topic-based Approach** This involves applying LDA for modelling topic distribution in the data sets, followed by the training of an SVM-based[1] classifier using the topic distribution as features.

Documents in the data set which will be introduced later are randomly and evenly spilt into training and test sets, keeping the ratio between relevant and irrelevant documents in each set the same as the ratio in the entire data set. Henceforth in this thesis, the documents relevant to a topic (i.e. positively labelled instances) are referred to as "relevant instances".

3.1.1 Pre-process

Documents in the data sets were first prepared for automatic classification using a series of pre-processing steps after tokenization. The pre-process consisted of stop-word removal, conversion of words to lower case and removal of punctuation, digits and words that appear only once. Finally, word counts were computed and saved in a tab-delimited format (SVMlight format)[2], for subsequent utilisation by the SVM classifiers.
Meanwhile, an ATR tool, TerMine[3], was used to identify multi-word terms in each document, as the basis for characterising their content. Preliminary experiments indicated that using only multi-word terms to characterise documents may not be sufficient since, in certain documents, the number of such terms could be small or zero. Accordingly, words and terms were retained as features for an independent experiment.

[1] SVM produces better result than others
[2] A file format representing features and their values. See Figure 3.1.1
[3] www.nactem.ac.uk/software/termine

3.1.2 Baseline Approach Design

The basic elements and methods for automatic text classification were introduced in Chapters 1 and 2. Here, the aim is to identify the best classifier and the relevant settings, including documents encoding methods, kernel functions and so on in order to create a robust baseline approach. Figure 3.2 demonstrates that the structure of the target baseline approach contains five crucial modules. **Pre-process** explained above forms the basis of the following four modules, dramatically reducing the noise in the corpus and accelerating the subsequent processes. The second module, **Building Dictionary** tokenizes each document in the corpus[4] and then learns the words (token) and multi-word terms that appear in the corpus. Each word obtained from documents is assigned a unique numeric ID for convenience. Module **Documents Encoding** transfers each document into a vector space which can be easily saved in a SVMlight file, according to a certain type of encoding scheme[5]. The final product of this module can be considered as a matrix with dimension $M \times W$ where M and W are the numbers of documents in the corpus and words in the dictionary respectively. The encoding schemes including Term Frequency (TF), Term Frequency/Inverse Document Frequency (TF/IDF) and Binary encoding were performed for further evaluation. Module **Training Classifier** receives the training data set from the corpus as an input and produces trained classifiers. The classifiers mentioned in Chapter 2 were trained excluding the Perceptron (NBC, K-NN and SVM). In module **Evaluation**, the different settings are evaluated and the best setting is chosen for the baseline approach. The setting in this section means which encoding scheme and classifier will be finally adopted.
The combination of TF/IDF encoding scheme and SVM was chosen as the baseline, since they outperformed all other combinations according to the results. The measures for evaluation will be discussed in Section 3.1.4.

3.1.3 Topic-based Approach Design

The topic-based approach applies LDA to produce a topic distribution for each document instead of the ordinary documents encoding schemes. LDA can be simply considered as a black box that receives an unlabelled corpus and a parameter K to indicate the number of topics. This black box outputs two tables, as shown in Figure 3.3, one

[4] corpus = training data + test data
[5] See Section 1.2.2

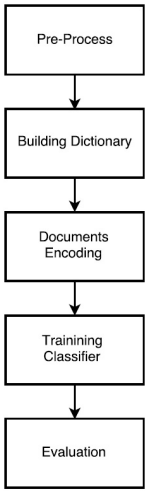

Figure 3.2: Workflow For The Baseline Approach

indicates the topic proportion for each document, and the other shows the probability that a word belongs to a topic (word clusters). The experiment applies LDA for modelling topic distribution in the data sets, followed by the training of an SVM-based classifier using the topic distribution as features.

A third-party implementation of LDA in python, Gensim [ŘS10], was used to predict the topic distribution for each document. This implementation uses a different method to estimate topics, an online algorithm[6]. Meanwhile, a Gibbs sampler was developed in order to compare these two estimators. However, since the results failed to show any significant difference, further details of comparison are not presented and the results produced by the Gibbs sampler will be used for the following sections. Finally, an evaluation will be performed on both the baseline approach and the topic-based approach in order to obtain a comprehensive conclusion.

3.1.4 Evaluation

To evaluate the classifiers, the standard metrics of precision, recall, F_1-measure[7], accuracy, area under the receiver operating characteristic curve (ROC) and precision-recall curve (PRC) are used. However, in this study, accuracy was found not to be a suitable indicator of effective performance, due to the significant imbalance between relevant

[6]The inference process receiving data piece by piece for saving RAM space

[7]β choice is subjective

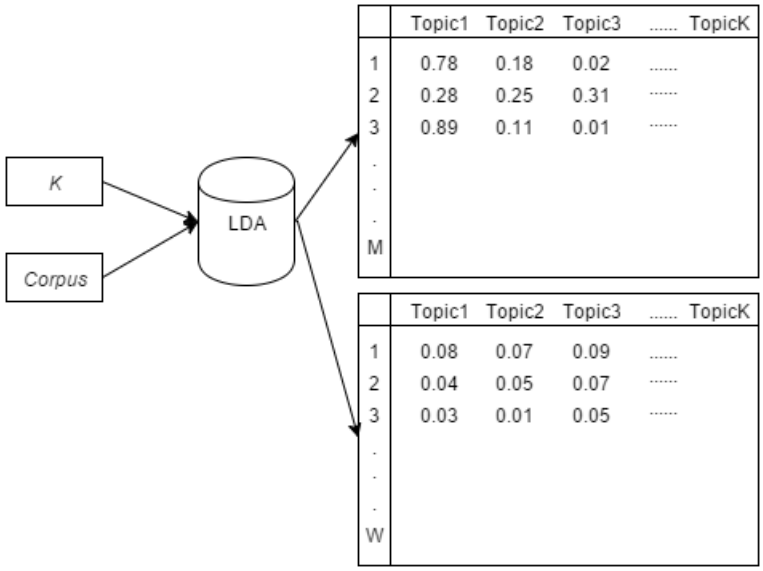

Figure 3.3: LDA Black Box

and irrelevant instances in the data set; this ratio is approximately 1:9 for each corpus, as shown in Table 3.1. Based on this ratio, weights are added to every training instance in order to reduce the influence caused by imbalanced data. [SKP06]. In evaluating classification performance, a particular emphasis is placed on recall since, as explained above, high recall is vital to achieve inclusiveness, which is considered to be an important factor in the perceived validity of a systematic review.

Table 3.1: Corpus information

	Positive Insts.	Total Insts.	Ratio	Feature Used	Type
Youth Development	1440	14538	0.099	title + abstract	social science
Cigarette Packaging	132	3156	0.041	title + abstract	social science
COPD	196	1606	0.122	title + abstract	clinical trial
Cooking Skill	197	9439	0.021	text	social science
Proton Beam	243	4751	0.051	title + abstract	clinical trials

3.1.5 Locating Terms

The multi-word terms were located during the pre-processing. In this section, the reason why this step is important and how it works will be given.
Since most of the corpora are domain-specific, non-compositional multi-word terms

may lose their original meaning if we split them into constituent words and ignore word order and grammatical relations. Thus, multi-word terms are automatically extracted using TerMine, which is a tool designed to discover them by ranking candidate terms from a part-of-speech (POS) tagged corpus according to C-value [FAM00]. Candidate terms are identified and scored via POS filters (e.g. adjective*noun+). A sub-set of these terms is extracted by defining a threshold for the C-value. TerMine makes use of both linguistic and statistical information in order to identify technical terms in a given corpus with the maximum accuracy. Other topic models attempt to present multi-word expressions in topics; for example, the LDA Collocation model [SG05] introduced a new latent variable to indicate if a word and its immediate neighbour can constitute a collocation. Unlike other methods, the advantage of TerMine is that it is applied independently of the topic modelling process. Thus, once it has been used to locate terms in a corpus, different topic models can be applied, without having to re-extract the terms each time the parameters of the topic model are changed. It is also important to note that long terms may have other shorter terms nested within them. Such nested terms may also be identified by TerMine. For example, "logistic regression model" contains the terms "logistic regression" and "regression model". However, there is no doubt that the original term "logistic regression model" is more informative. Thus, the strategy to locate the terms is that the longer terms are given higher priority to be matched and our maximum length for a term is four tokens. The identified terms will be treated as a single word by adding a special character like a dash between each token. For example, "computer science" will be changed to "computer_science", which can be recognised by the tokenization program.

3.1.6 Parameter Settings

As for parameter tuning, all the experiments have been performed with default parameters for classifiers and symmetry hyperparameters for LDA, which means that every topic will be sampled with equal probability.

3.2 Results and Discussion

The experiments were performed using five data sets corresponding to completed reviews, in domains of social science and clinical trials. These reviews constitute "gold standard" data, in that for each domain, they include expert judgements about which

documents are relevant or irrelevant to the study in question. The data sets were used as the basis for the intrinsic evaluation of the different text classification methods. Our conclusions are supported by the Friedman test (Table 3.2) which is a non-parametric test that measures how different the performance of 5 data sets are, based on ranking. Given that the methods applied produced roughly comparable patterns of performance across each of the five different data sets, this thesis reports only on the performance for one of the corpora.

Table 3.2: Friedman Test for 5 data sets on different kernel functions and documents representation.(BOW: Bag-of-word features, TPC: Topic features, TE: Term-enriched topic features, RBF: Radial basis function kernel POLY: Polynomial kernel)

	Linear			RBF			POLY		
	BOW	TPC	TE	BOW	TPC	TE	BOW	TPC	TE
Precision									
Mean Rank	2.90	2.00	1.10	1.00	2.50	2.50	1.2	2.6	2.2
$P =$	0.0001			0.00196			0.001501		
Recall									
Mean Rank	1.00	2.60	2.40	1.00	2.40	2.60	1.20	2.40	2.40
$P =$	0.00332			0.0256			0.008977		
F-score									
Mean Rank	2.60	2.10	1.30	1.00	2.60	2.40	1.20	2.60	2.20
$P =$	0.08977			0.00332			0.01501		
ROC									
Mean Rank	3.00	1.80	1.20	1.00	2.60	2.40	1.00	2.60	2.40
$P =$	0.00066			0.00332			0.00332		
PRC									
Mean Rank	2.80	2.00	1.20	1.00	2.70	2.30	1.00	2.60	2.40
$P =$	0.0168			0.0008			0.84935		

3.2.1 Data Set

The models are applied to three data sets provided by the Evidence Policy and Practice Information and Coordinating Centre (EPPI-center)[8] and two data sets previously

[8]http://eppi.ioe.ac.uk/cms/

presented in Wallace et al. [WSBT10]. These labelled corpora include reviews ranging from clinical trials to reviews in the domain of social science. The data sets correspond specifically to Cigarette Packaging, Youth Development, Cooking Skills, COPD (Chronic obstructive pulmonary disease), Proton Beam and Hygiene Behaviour (Table 3.1). Each corpus contains a large number of documents and, as mentioned in Section 3.1.4, there is an extremely low proportion of relevant documents in each case. For example, the Youth Development corpus contains a total of 14,538 documents, only 1,440 of which are relevant to the study. The Cigarette Packaging sub-set contains 3,156 documents in total, with 132 having been marked as relevant.

3.2.2 BOW-based classification

Table 3.3: Evaluation on all corpora of SVM classifiers trained with TF-IDF features

	Precision	Recall	F_1-score	Accuracy	ROC	PRC
YOUTH DEVELOPMENT						
Linear	0.394	0.686	0.501	0.862	0.891	0.508
RBF[1]	0.0	0.0	0.0	0.899	0.131	0.055
POLY[2]	0.0	0.0	0.0	0.899	0.153	0.054
CIGARETTE PACKAGING						
Linear	0.367	0.707	0.484	0.937	0.939	0.477
RBF	0.0	0.0	0.0	0.958	0.063	0.021
POLY	0.0	0.0	0.0	0.958	0.082	0.021
COOKING SKILL						
Linear	0.366	0.482	0.416	0.967	0.922862	0.328
RBF	0.0	0.0	0.0	0.975	0.079	0.012
POLY	0.0	0.0	0.0	0.975	0.512	0.500
COPD						
Linear	0.595	0.773	0.672	0.909	0.927	0.720
RBF	0.0	0.0	0.0	0.879	0.066	0.064
POLY	0.0	0.0	0.0	0.879	0.113	0.067
PROTON BEAM						
Linear	0.057	0.078	0.066	0.881	0.562	0.063
RBF	0.0	0.0	0.0	0.9465	0.442	0.048
POLY	0.0	0.0	0.0	0.946	0.482	0.054

[1] **RBF:** Radial basis function kernel.
[2] **POLY:** Polynomial kernel.

Table 3.3 shows the performance of the SVM classifiers trained with TF-IDF features when applied to all corpora. Due to the imbalance between relevant and irrelevant instances in the data set, each positive instance was assigned a weight, as mentioned above. Default values for SVM training parameters were used (i.e. no parameter tuning was carried out), although three different types of kernel functions were investigated, i.e. linear, radial basis function (RBF) and polynomial (POLY). Unlike the linear kernel that aims to find a unique hyperplane between positive and negative instances, RBF and POLY can capture more complex distinctions between classes than the linear kernel. As illustrated in Figure 3.4, the BOW-based classification achieved the best performance when the linear kernel function is used. However, it is necessary to recall that the ratio of positively (i.e. relevant) to negatively (i.e. irrelevant) labelled instances is approximately 1:9 in our corpora. Hence, even if a classifier labels all test samples as irrelevant instances, a very high accuracy will still be obtained. However, for systematic reviews, it is important to retrieve the highest possible number of relevant documents, so recall is a much better indicator of performance than accuracy. Secondly, both the RBF and POLY kernel functions obtained zero for precision, recall and F_1-score. This can be attributed to the imbalanced nature of the corpora [AKJ04]. Additionally, the BOW representation produces a high dimensional space (given the large number of unique words in the corpora). In this high dimensional space, the two non-linear kernels (RFB and POLY) yielded a very low performance.

3.2.3 Topic-based Classification

Topic-based classification was undertaken by first analysing and predicting the topic distribution for each document and then classifying the documents using topics as features. During the phase of training the model, the topic assigned to each word in a document can be considered as a hidden variable, problems that involve hidden variables can be tackled by using approximation methods such as Monte Carlo Markov Chain (MCMC) or variational inference. However, these methods are sensitive to initial parameter settings which are usually set randomly before the first iteration. Consequently the results could fluctuate within a certain range as mentioned in Section 1.3.2. The results produced by topic-based classification are all average results. However, our results show that topic distribution is an ideal replacement for the traditional BOW features. Besides other advantages, the most obvious advantage is to reduce the dimensions of features for representing a document. Experimental settings were identical in

Table 3.4: Evaluation on the youth development data set of SVM classifiers trained with topic features

						LINEAR
Topic density[1]	Precision	Recall	F_1-score	Accuracy	ROC	PRC
2	0.156	0.767	0.281	0.552	0.685	0.169
5	0.163	0.769	0.280	0.612	0.7485	0.215
10	0.216	0.775	0.334	0.706	0.782	0.239
20	0.228	0.776	0.316	0.662	0.778	0.276
30	0.235	0.772	0.357	0.732	0.816	0.288
40	0.239	0.773	0.364	0.732	0.820	0.320
50	0.232	0.767	0.356	0.722	0.818	0.371
60	0.249	0.777	0.353	0.717	0.811	0.338
70	0.283	0.771	0.407	0.789	0.841	0.342
80	0.279	0.782	0.354	0.782	0.842	0.359
90	0.280	0.774	0.376	0.748	0.832	0.345
100	0.281	0.786	0.376	0.751	0.831	0.358
150	0.290	0.791	0.379	0.740	0.836	0.367
200	0.294	**0.791**	0.423	0.770	**0.850**	**0.408**
300	**0.322**[2]	0.720	**0.4558**	0.958	0.847	0.389
500	0.305	0.708	0.427	0.80	0.844	0.395
						RBF
2	0.151	0.812	0.254	0.522	0.694	0.168
5	0.159	**0.8260**	0.266	0.544	0.719	0.194
10	0.189	0.802	0.306	0.635	0.775	0.201
20	0.199	0.745	0.314	0.674	0.774	0.253
30	0.257	0.679	0.373	0.770	0.816	0.312
40	0.264	0.620	0.371	0.788	0.7992	0.301
50	0.246	0.641	0.356	0.767	0.779	0.250
60	0.235	0.576	0.334	0.769	0.778	0.250
70	0.255	0.490	0.335	0.805	0.773	0.237
80	0.395	0.390	**0.382**	0.873	0.806	0.318
90	**0.409**	0.219	0.285	0.889	0.801	0.312
100	0.368	0.019	0.036	0.898	**0.817**	**0.319**
150	0	0	0	0.877	0.812	0.297
						POLY
2	0.153	**0.826**	**0.258**	0.523	0.702	**0.170**
5	**0.171**	0.143	0.156	0.843	**0.704**	0.166
10	0	0	0	0.899	0.285	0.060

[1] Results are reported according to different values of the topic density.
[2] Items in bold refer to the highest score obtained in a column.

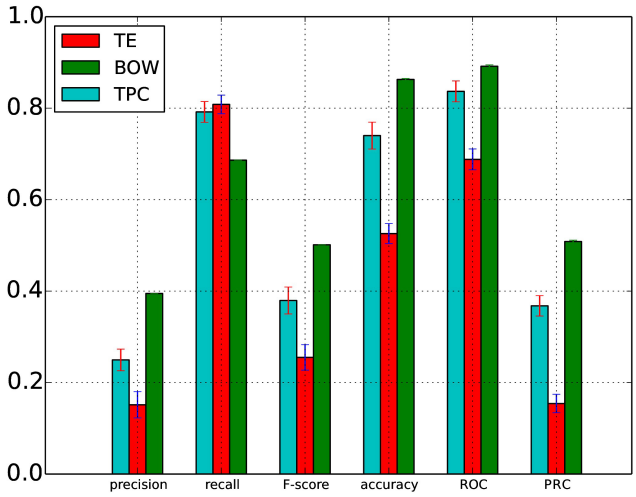

Figure 3.4: **Linear kernel function.** Comparison between the performance of BOW-based, topic distribution-based and term-enriched topic classifiers trained using a linear kernel function.

the evaluation of the two sets of classifiers, except for the features being topic distributions in the one case and BOW in the other. The optimal LDA model was derived through experimentation with differing numbers of topics (which can also be referred to as "topic density"). In the experiments performed, several values for this parameter were explored. Table 3.4 shows the results of the evaluation of SVM models trained with topic distribution features using linear, RBF and POLY kernel functions, respectively. The table also shows how the performance varies according to different topic density values for the LDA model. These values were varied from 2 to 100 (inclusive), in increments of 10, and from 100 to 500 in increments of 100. Generally, each topic density would correspond to a certain size of corpus and vocabulary. Empirically, the larger the size of corpora and vocabulary, the greater the number of topics that is needed to accurately represent their contents, and vice versa. Tables 3.5 and 3.6 show two samples of sets of words and/or terms that are representative of a topic in the same corpus (youth development). Term-enriched (TE) topics include multi-word terms identified by TerMine as well as single words, whilst ordinary topics consist only of single words. From the tables, it can be clearly seen that TE topics are more distinctive and readable than single-word topics. As the classification performance was similar to the single-word topic-based classification, a table like Table 3.4 will not be presented here. However, a comparison of the classification performance for the three

approaches i.e. BOW-based, topic-based and TE-topic-based will be presented in the next section.

3.2.4 Comparison Of Approaches

A comparison of the performance of the BOW-based model (BOW in legend) against the performance of models trained with topic-based model (TPC) and term enriched-topic model (TE) is presented in this section. According to the results of using a linear function for model training (Figure 3.4), models based on topic and TE-topic distribution features yield lower precision, F-score, ROC and PRC, but obtain higher recall. The best performing topic-based model (with topic density set to 150 for Youth Development corpus) was used. For this comparison, the best performing topic-based model (with topic density set to 150 for Youth Development corpus) was used. It can be observed from Figure 3.4 that the BOW-based model outperforms the topic and TE-topic based one in terms of all metrics except for recall. Figures 3.5 and 3.6 illustrate the results of using RBF and POLY kernel functions, respectively, in training BOW, topic-based models and TE-topic-based model on the Youth Development Corpus. It can be observed that employing these kernels, the SVM models trained with topic and TE-topic distributions outperform those trained with BOW features by a large margin. Another observation is that training using RBF and POLY kernel functions significantly degraded the performance of BOW-based models.

Using RBF and POLY kernel functions, the BOW-based classifiers perform poorly, with zero in precision, recall and F-score. As noted earlier, high accuracy is not a good basis for judging performance due to the imbalance between positive and negative instances, i.e. even if a classifier labels every document as a negative sample, accuracy will still be around 90%. Figure 3.7 gives the comparison of different kernel functions using topic features on the Youth Development corpus, indicating that taking all measures into account, a linear kernel function gave the best overall performance, achieving the highest score in every metric other than recall. However, both RBF and POLY kernel functions outperformed linear, albeit by only 4%, on the recall measure, which we have identified as highly pertinent to the systematic review use-case. This study uses a generic list of kernel functions ranked from high to low in terms of recall for topic-based and TE-topic-based feature: POLY>RBF>LINEAR. For a ranked list of feature types in terms of recall, it is: TPC>TE>BOW. Additionally, Precision-Recall and ROC curves achieved by the models are presented in Appendix A.3.

Table 3.5: Term-enriched topics

topic 1	topic 2	topic 3
school	**teen birth rates**	**program activity**
plains	school	murders
murders	weakly	educare
cultural tradition	corresponds	projected
gang membership	ngos	**multidimensional index**
juvenile delinquency prevention program	chile	**program activity**
immigration	**latino culture**	**fast track**
educare	wore	**socio-economic circumstance**
recollections	**nonneglected children**	**nonneglected children**
program activity	skillful	hopkins

Table 3.6: Ordinary topics

topic 1	topic 2	topic 3
forged	bosnian	horizons
school	acculturationrelated	pascd
educare	revitalization	steps
nonconcordant	chipce	healthier
nonfarmers	api	wore
eightythree	unavailability	fibrosis
mdma	paradigmatic	eurocentric
privatized	individualist	justified
chile	phonics	noncollege
discontinue	fulfils	correspond

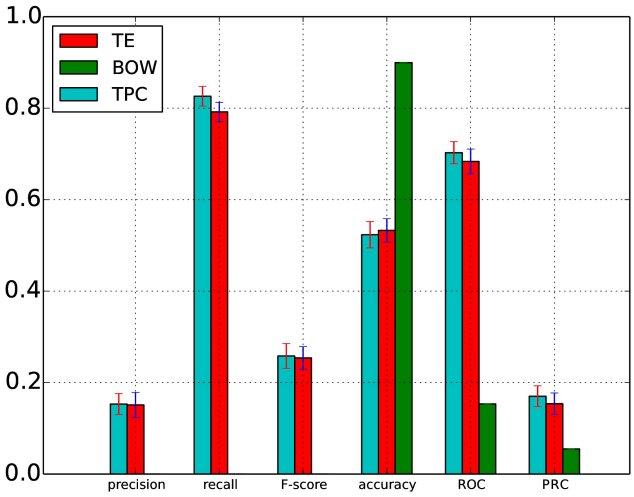

Figure 3.5: **RBF Kernel function.** Comparison between the performance of BOW-based, topic distribution-based and term-enriched topic classifiers trained using a RBF kernel function.

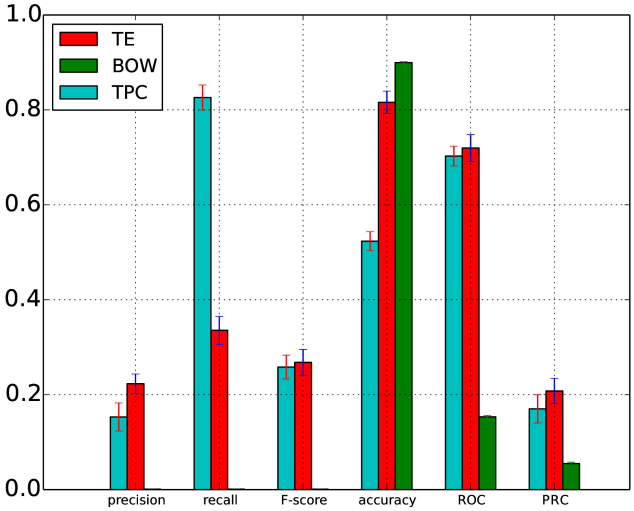

Figure 3.6: **POLY kernel function.** Comparison between the performance of BOW-based, topic distribution-based and term-enriched topic classifiers trained using a POLY kernel function. .

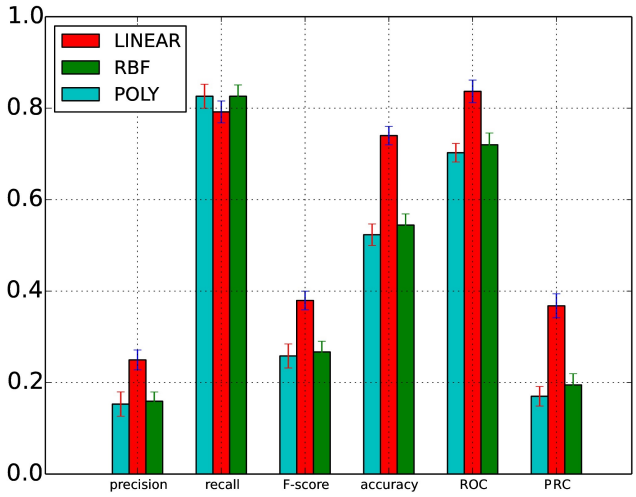

Figure 3.7: **Different kernel functions.** Comparison between the performance of Linear, RBF and POLY kernel functions using topic feature.

Chapter 4

Conclusion

4.1 Summary

Systematic reviews can produce the most reliable results for research, while automatic text classification could dramatically reduce the burden of reviewers in the screening phase which is the most important part in systematic reviews. This has been proved by the previous studies. However, traditional automatic text classification requires a BOW model that represents a document as a set of words ignoring grammar and the order of the words. Documents represented in such form would have many drawbacks. First of all, documents in such high dimensional space are normally not seen by the reviewers. Secondly, high-dimensional space would decrease the performance of classifiers. These two factors would have a significant impact on the screening phase of systematic reviews.

LDA simulates a much more natural way for composing an article (document). It assumes there are many existing topics before creating any document. Each topic is a cluster of words. In order to generate a document, a set of topics needs to be decided first, and then these clusters (topics) will be used for generating words like rolling a die. This is a generative process for generating a collection of documents. However, the realistic situation is the reverse, i.e. given a collection of documents, estimate the topics. By using approximation methods such as Gibbs Sampler for LDA, we are able to obtain the topics and allowed to represent a document in a more abstract way instead of BOW.

Our experiments aimed to carry out a systematic comparison between the classifiers using traditional BOW representation and the LDA representation. In order to achieve this, the first step was to identify a robust baseline. Many classifiers and encoding

methods have been tested and finally the combination of TF/IDF + SVM has been decided as the best setting for BOW model which is considered as the baseline method. The second step was to evaluate the performance of the combination LDA + SVM, where the topics of each document estimated by LDA would be used as input features for SVM. Finally, the results produced by using the baseline method and LDA-based approach were compared under the same measures.
The experiments demonstrated that the performance of BOW SVM with linear kernel function produced the most robust results achieving the highest values in almost every metric, except for recall. However, for any systematic review classification task, poor performance in recall needs to be addressed. The BOW model yielded a poor performance with RBF and POLY kernel functions due to the data imbalance and dimensionality issue. Topic-based classification significantly addresses this problem by dramatically reducing the dimensionality of the representation of a document (topic feature). The topic-based classifier yielded a higher recall, which means more relevant documents will be identified. Moreover, the topic features enable the classifier to work with RBF and POLY kernels and produce better recall comparing with a linear kernel. The same patterns were observed in all corpora, although there is only one example (Youth Development) presented in this thesis.

4.2 Future Work

This thesis addresses internal validity threats of automatic citation screening methods. Threats to external validity of our method will be assessed by an extrinsic evaluation process that will be carried out as part of our future work.
Also, future work will further investigate the generalisability of the model to diverse domains. Moreover, different machine learning and text mining techniques that can be used to support systematic reviews such as paragraph vectors, active learning will be explored.
This work focuses on an intrinsic evaluation of automatic screening methods while as future work, we will conduct an extrinsic evaluation according to which the model will be used in live systematic reviews. Further experiments will be performed in a more realistic situation. For example, whether topics could help reviewers' decision in 'live' systematic review would be an interesting research area in the future. An intuitive picture of Term-Enriched topics has been made in this article. For public health reviews where topics are multidimensional, the presence of diverse multi-word terms in a data

set can be an important element that affects the performance of classifiers. However, TE-topics have the potential to deal with these difficulties. Further investigation on TE-topics will be performed, which would benefit reviewers and help them to understand topics more easily compared to ordinary topics that contain only single words.

References

[AACZ14] J.J. García Adeva, J.M. Pikatza Atxa, M.U. Carrillo, and E.A. Zengotitabengoa. Automatic text classification to support systematic reviews in medicine. *Expert Systems with Applications*, 41(4):1498–1508, 2014.

[AGM12] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond SVD. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10, New Brunswick, NJ, 2012. IEEE.

[AKJ04] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, Berlin Heidelbeg, 2004.

[ARO+09] Sophia Ananiadou, Brian Rea, Naoaki Okazaki, Rob Procter, and James Thomas. Supporting systematic reviews using text mining. *Social Science Computer Review*, 27:509–523, 2009.

[BDF10] Tanja Bekhuis and Dina Demner-Fushman. Towards automating the initial screening phase of a systematic review. *Studies in Health Technology and Informatics*, 160(Pt 1):146–150, 2010.

[BDF12] Tanja Bekhuis and Dina Demner-Fushman. Screening nonrandomized studies for medical systematic reviews: A comparative study of classifiers. *Artificial Intelligence in Medicine*, 55(3):197–207, 2012.

[BDO95] M.W. Berry, S.T. Dumais, and G.W O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[BNB+10] Florian Boudin, Jian-Yun Nie, J.C. Bartlett, Roland Grad, Pierre Pluye, and Martin Dawes. Combining classifiers for robust PICO element

detection. *BMC Medical Informatics and Decision Making*, 10(1):29, 2010.

[BNJ03] D.M. Blei, Andrew Y Ng, and M.I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[BTL09] Michael Barza, Thomas A Trikalinos, and Joseph Lau. Statistical considerations in meta-analysis. *Infectious Disease Clinics of North America*, 23(2):195–210, 2009.

[BTMDF14] Tanja Bekhuis, Eugene Tseytlin, K.J. Mitchell, and Dina Demner-Fushman. Feature engineering and a proposed decision-support system for systematic reviewers of medical evidence. *PLoS ONE*, 9(1):e86277, 2014.

[Coh08] A.M. Cohen. Optimizing feature representation for automated systematic review work prioritization. In *AMIA annual symposium proceedings*, volume 2008, page 121. American Medical Informatics Association, 2008.

[Coh11] A.M. Cohen. Performance of support-vector-machine-based classification on 15 systematic review topics evaluated with the WSS@95 measure. *Journal of the American Medical Informatics Association*, 18(1):104–104, 2011.

[Cou97] Carl Counsell. Formulating questions and locating primary studies for inclusion in systematic reviews. *Annals of Internal Medicine*, 127(5):380–387, 1997.

[Cov68] T.M. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1):50–55, 1968.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[CWPY06] A.M. Cohen, R.H William, K. Peterson, and Po-Yin Yen. Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, 13(2):206–219, 2006.

[DFL07] Dina Demner-Fushman and Jimmy Lin. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics*, 33(1):63–103, 2007.

[Din08] I.D. Dinov. Expectation maximization and mixture modeling tutorial. *Statistics Online Computational Resource*, 2008. `http://escholarship.org/uc/item/1rb70972`.

[DS02] M.H. DeGroot and Mark J Schervish. *Probability and Statistics*. Addison Wesley, 2002.

[FAM00] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic recognition of multi-word terms:. the C-value/NC-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000.

[FBS+10] Marcelo Fiszman, B.E. Bray, Dongwook Shin, Halil Kilicoglu, G.C. Bennett, Olivier Bodenreider, and T.C. Rindflesch. Combining relevance assignment with quality of the evidence to support guideline development. *Studies in Health Technology and Informatics*, 160(1):709, 2010.

[FIM10] Oana Frunza, Diana Inkpen, and Stan Matwin. Building systematic reviews using automatic text classification techniques. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 303–311, Beijing, China, 2010. Association for Computational Linguistics.

[FSM13] K.R. Felizardo, Simone R.S. Souza, and J.C. Maldonado. The use of visual text mining to support the study selection activity in systematic literature reviews: A replication study. In *Replication in Empirical Software Engineering Research (RESER), 2013 3rd International Workshop on*, pages 91–100, Baltimore, MD, 2013. IEEE.

[GLWW00] Christoph Goller, Joachim Löning, Thilo Will, and Werner Wolff. Automatic document classification—A thorough evaluation of various methods. *Internationales Symposium Informationswissenschaft*, 2000:145–162, 2000.

[Gri02] Tom Griffiths. Gibbs sampling in the generative model of latent dirichlet allocation. Technical report, 2002. `https:`

//people.cs.umass.edu/~wallach/courses/s11/cmpsci791ss/readings/griffiths02gibbs.pdf.

[GS04] T.L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[Han12] Li Hang. *Sstatistical Learning Methods*. Qinghua University Press, 2012.

[HC06] Lawrence Hunter and K.B. Cohen. Biomedical language processing: what's beyond PubMed? *Molecular Cell*, 21(5):589–594, 2006.

[Hei05] Gregor Heinrich. Parameter estimation for text analysis. Technical report, 2005. http://www.arbylon.net/publications/text-est.pdf.

[HER09] Keith Henderson and Tina Eliassi-Rad. Applying latent dirichlet allocation to group discovery in large graphs. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1456–1461. ACM, 2009.

[HGA+11] J.P. Higgins, Sally Green, P Alderson, M Clarke, C.D. Mulrow, and A.D. Oxman. *Cochrane Handbook for Systematic Reviews of Interventions, Version 5.1.0*, volume 5. The Cochrane Collaboration, 2011.

[Hof99] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[Joa98] Thorsten Joachims. *Text Categorization with Support Vector Machines: Learning with many relevant features*. Springer, 1998.

[JP13] Siddhartha Jonnalagadda and Diana Petitti. A new iterative method to reduce workload in systematic review process. *International Journal of Computational Biology and Drug Design*, 6(1):5–17, 2013.

[KL51] Solomon Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[LKE08] S.K. Lukins, Nicholas Kraft, and L.H. Etzkorn. Source code retrieval for bug localization using latent dirichlet allocation. In *15th Working Conference on Reverse Engineering, 2008. WCRE'08.*, pages 155–164. IEEE, 2008.

[LLB08] Erik Linstead, Cristina Lopes, and Pierre Baldi. An application of latent dirichlet allocation to analyzing software evolution. In *Seventh International Conference on Machine Learning and Applications, 2008. ICMLA'08.*, pages 813–818. IEEE, 2008.

[LM14] Moontae Lee and David Mimno. Low-dimensional embeddings for interpretable anchor-based topic inference. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1319–1328. Association for Computational Linguistics, 2014.

[MKI+10] Stan Matwin, Alexandre Kouznetsov, Diana Inkpen, Oana Frunza, and Peter O'Blenis. A new algorithm for reducing the workload of experts in performing systematic reviews. *Journal of the American Medical Informatics Association*, 17(4):446–453, 2010.

[ML02] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2012.

[MSH08] Girish Maskeri, Santonu Sarkar, and Kenneth Heafield. Mining business topics in source code using latent dirichlet allocation. In *Proceedings of the 1st India software engineering conference*, pages 113–120. ACM, 2008.

[MTOEA14] Makoto Miwa, James Thomas, Alison OMara-Eves, and Sophia Ananiadou. Reducing systematic review workload through certainty-based screening. *Journal of Biomedical Informatics*, 51:242–253, 2014.

[Nea00] R.M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[Nov63] A.B. Novikoff. On convergence proofs for perceptrons. In *Proceedings of the Symposium on Mathematical Theory of Automata*, pages 615–622, New York, 1963.

[OETM+15] Alison O'Mara-Eves, James Thomas, John McNaught, Makoto Miwa, and Sophia Ananiadou. Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic reviews*, 4(1):5, 2015.

[OFMF15] F.R. Octaviano, K.R. Felizardo, J.C. Maldonado, and S.C.P.F. Fabbri. Semi-automatic selection of primary studies in systematic literature reviews: is it reasonable? *Empirical Software Engineering*, 20(6):1898–1917, 2015.

[PSD00] J.K. Pritchard, Matthew Stephens, and Peter Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

[PTRV98] C.H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168. ACM, 1998.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[ŘS10] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[RU12] Anand Rajaraman and J.D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2012.

[SC14] Gagan Sidhu and Brian Caffo. MONEYBaRL: Exploiting pitcher decision-making using reinforcement learning. *The Annals of Applied Statistics*, 8(2):926–955, 2014.

[SG05] M. Steyvers and T. Griffiths. Matlab topic modeling toolbox 1.3., 2005. http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm.

[SKP06] Kotsiantis Sotiris, Dimitris Kanellopoulos, and Panayiotis Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30(1):25–36, 2006.

[Soe85] Dagobert Soergel. *Organizing information: principles of data base and retrieval systems*. Elsevier, 1985.

[WSBT10] B.C Wallace, Kevin Small, C.E. Brodley, and T.A. Trikalinos. Active learning for biomedical citation screening. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 173–182. ACM, 2010.

[WTL+10] B.C. Wallace, T.A. Trikalinos, Joseph Lau, Carla Brodley, and C.H. Schmid. Semi-automated screening of biomedical citations for systematic reviews. *BMC bioinformatics*, 11(1):55, 2010.

Appendix A

Appendix

A.1 Abbreviations

BOW: Bag-of-words;
LDA: Latent Dirichlet Allocation;
ATR: Automatic Term Recognition;
PICO: the Population, the Intervention, Comparator and the Outcome;
SVM: Support Vector Machine;
SVD: Singular Value Decomposition;
KNN: K-nearest Neighbour;
TF-IDF: Term Frequency/Inverse Document Frequency;
ROC: Receiver Operating Characteristic;
PRC: Precision-recall Curve;
POS: Part-of-speech;
RBF: Radial Basis Function;
POLY: Polynomial;
MCMC: Monte Carlo Markov Chain;
TE: Term-enriched;
NBC: Naive Bayes Classifier;
EM: Expectation Maximisation;
MLE: Maximum Likelihood Estimators.

A.2 Publication

Part of the work presented in this thesis has been published in the following peer-reviewed journal article:
Yuanhan Mo, Georgios Kontonatsios and Sophia Ananiadou. Supporting systematic reviews using LDA-based document representations *Systematic Reviews 2015*, 4:172 (26 November 2015)

A.3 Supplementary Figures

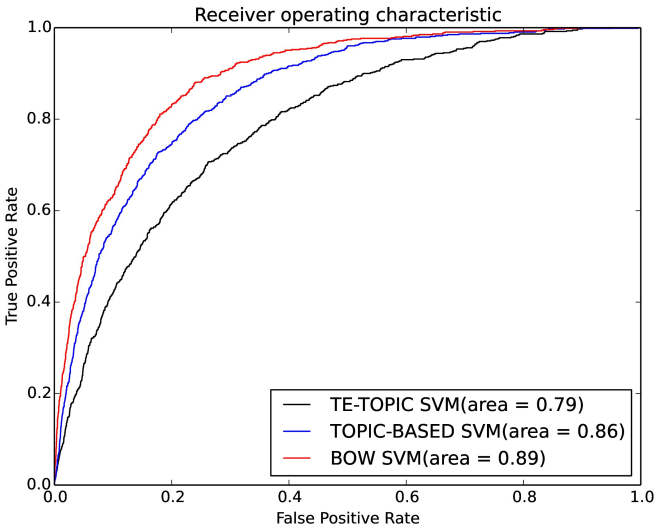

Figure A.1: **Receiver Operating Characteristic Curve:** linear kernel function.

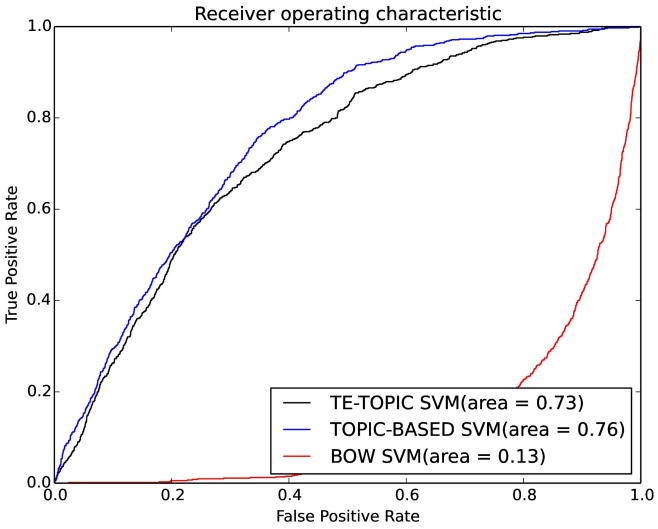

Figure A.2: **Receiver Operating Characteristic Curve:** RBF kernel function.

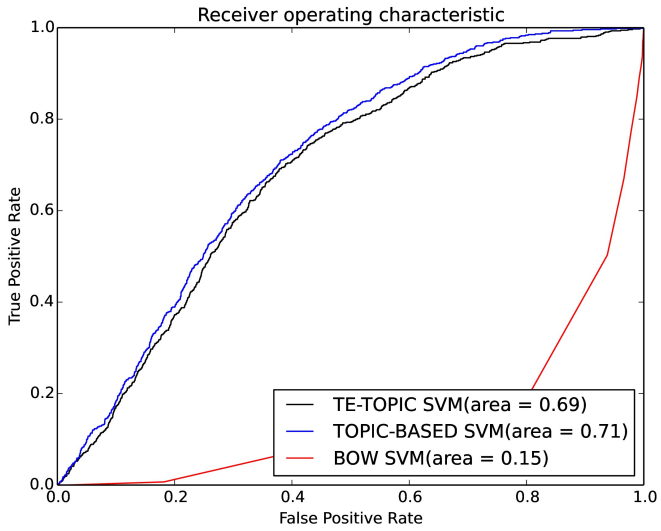

Figure A.3: **Receiver Operating Characteristic Curve:** Poly kernel function

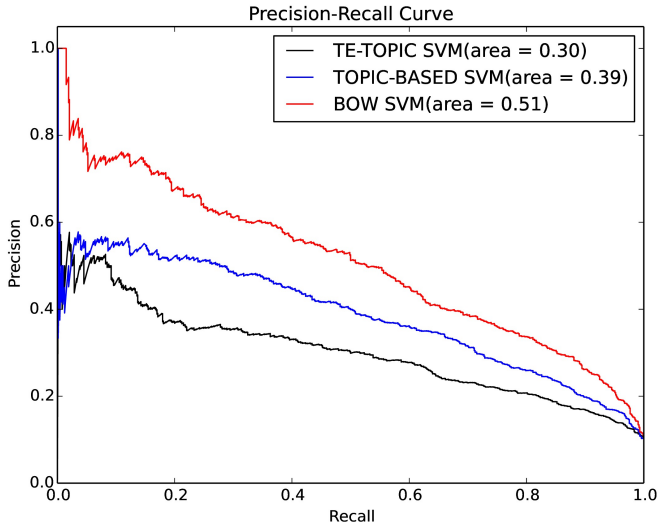

Figure A.4: **Precision-Recall Curve:**linear kernel function.

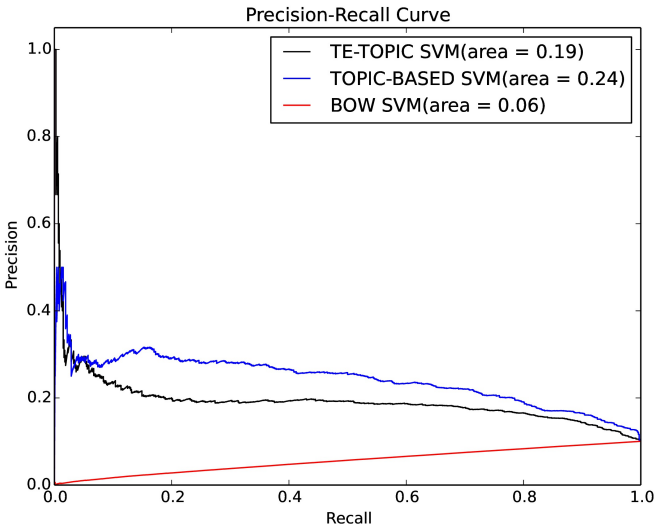

Figure A.5: **Precision-Recall Curve:**RBF kernel function.

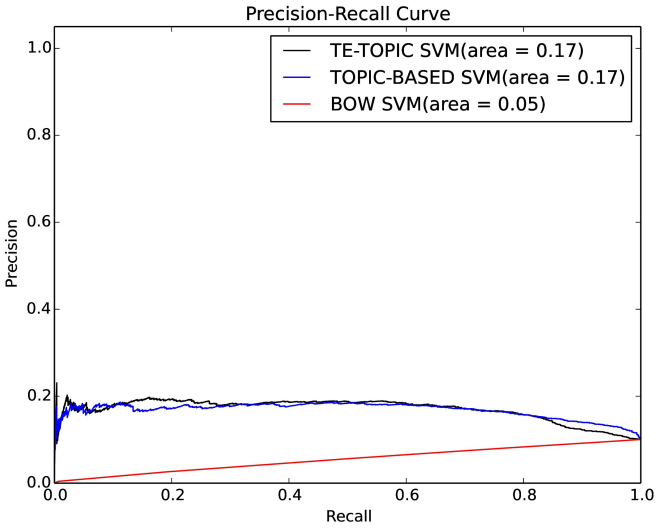

Figure A.6: **Precision-Recall Curve:**POLY kernel function.