Automatic evaluation of privacy policy
a machine learning approach

Sun, Y.

Award date:
2012

Automatic Evaluation of Privacy Policy

A Machine Learning Approach

by

Yuanhao Sun

Supervisor: Professor Milan Petković

Submitted in partial fulfilment of the requirements for the Degree of Master of Science
in the Department of Mathematics and Computer Science

Eindhoven University of Technology

Feburary 2012

Abstract

Privacy is taking an increasingly prominent place in today's digital world. People wish to control their private information when interacting with websites on the Internet. However, customer information is one of the keys to successful Internet business. Internet companies wish to gather and use as much customer information as possible in order to facilitate business, build competitive barrier and generate profits. In the middle of the conflicting notions, privacy policy serves as the main channel for the companies to disclose their practice for dealing with user's private information.

However, many surveys have shown that users seldom read the privacy policies and the current mechanisms to present website privacy policies have not been successful. The readability issue of privacy policies calls for automated ways of privacy policy evaluation to assist users to quickly gain insights about the privacy practice of the website.

This research addresses the present gap in the communication and understanding of privacy policies, by creating an automated privacy policy evaluation framework that provides automatic categorization, analysis and grading of privacy policies. We advocate a machine learning approach towards privacy policy evaluation and lay the fundamental basis for this new approach.

We present a privacy policy evaluation framework. The framework comprises several core components, such as privacy policy paragraph categorization, privacy policy grading, share statement understanding, as well as some accessorial components, such as privacy policy detection, search result extraction, text extraction, visualization, and web application interface. We define the common categories of privacy policies, label real-world privacy policies to form the datasets and implement all the aforementioned components.

We investigate the application of text classification to categorize privacy policy paragraphs. We present extensions to this categorization scheme, such as two-layered classification, multi-label classification, grading and visualization. We propose two approaches for the task of share statement understanding, each with different variants. We extensively experiment with our proposed schemes and approaches on the privacy policy datasets. Our study results demonstrate that the machine learning approach is effective in building privacy policy evaluation systems that serve the purpose to assist users to better understand the privacy policies.

Forward

This thesis marks the end of the exciting and enjoyable journey to achieve my master degree at Eindhoven University of Technology. It is the result of my graduation project, which was performed internally at the Security group of the Mathematics and Computer Science department.

This thesis concludes the major effort in my graduation project, which is applying machine learning and natural language processing techniques to provide new solutions to real-life privacy problems. I believe this is what I do best and where my passion lies – bringing knowledge across areas in order to solve real-life problems.

This work would not have been possible without the help and support from a number of people. Milan Petković was the best supervisor I could have hoped for. He gave me the freedom and encouragement to pursuit the ideas I had, but also pointed me to a right direction when necessary. He provided insightful instructions on both the project work and my intellectual development. I am truly grateful for having the chance to work under his supervision.

Fully thankful for all the helps and comments, I would like to express my great gratitude to Elisa Costante, who was my thesis tutor, Mykola Pechenizkiy and Jerry den Hartog. This work would not exist without Elisa's initial idea in applying text categorization to privacy policy. The suggestions from Mykola brought valuable machine learning expertise to this project.

I am also very appreciative of my parents and my girlfriend, Xue, for their support and encouragement. Finally, I would like to thank all my friends around me, physically or virtually, for making my life colorful.

Yuanhao Sun
January 2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Privacy is important and valuable for online users. In a recent survey[1], 90% of 5,778 respondents (from the EU countries and the U.S.) feel that their personal information, reputation and privacy are at risk on the Internet today. In another recent survey [84], 94% of 1004 consumers (from the U.S.) consider online privacy important. Consumers are willing to pay a premium if an e-commerce website presents a prominent display of user-friendly privacy practices [104].

Even users care about their private information, the privacy policies do not influence the user's trust perception, mainly because users do not read privacy policies [21]. A study under laboratory conditions shows that only 26% participants read privacy policies and it is also believed that readership outside of laboratory conditions is far lower [45]. Users do not read privacy policies due to the poor readability. An online survey [62] of more than 700 participants tested privacy policies in three formats and finally found that "participants were not able to reliably understand companies privacy practices with any of the formats" and that "all formats and polices were similarly disliked". Most privacy policies are written at a level that requires a college-level education and use specific domain terminology that users are not familiar with [44, 62, 37].

Full text privacy policies in natural language form are still the de facto standard for presenting privacy policy information online [50], though there are many solutions proposed to better present the privacy policies. Examples of these altern-

[1] http://www.slideshare.net/123people/123people-privacy-survey-final – 123people online privacy survey results for Data Privacy Day on January 28th 2011.

atives are the Platform for Privacy Preferences (P3P) [107], layered privacy notices [30, 29], standardized table and short standardized table [50] and nutrition label [49]. A common problem for all these alternatives is that their success heavily rely on the adoption and co-operation from the websites.

There are two conflicting notions that create a gap between the privacy policy research and real-world practice. On the one hand, current natural language privacy policies have a widely criticized readability issue, due to the use of long texts and complex legalistic phrases. This is the main reason why reading privacy policies is both challenging and time consuming. On the other hand, it is unlikely that the eco-system in the industry will change dramatically in near future, given the natural language privacy policies are still prevailing after years of research efforts on alternative presentations of privacy policies.

In short, many surveys have demonstrated that online users are highly concerned about online privacy, yet current alternative mechanisms to natural language in presenting privacy policies have not achieved major successes due to the present gap between the current research directions and the real-world practice.

1.2 Research goals

In order to address the present gap, this master thesis project aims to apply a generic machine learning approach to build automatic privacy policy evaluation systems, which assist users in reading and understanding the privacy policies by making it less challenging and time consuming.

The first goal of this thesis work is to build a privacy policy paragraph categorization system as well as its extensions such as privacy policy grading and visualization. After this system is implemented, our objective is to test it on certain datasets in order to prove the generic machine learning approach is effective for privacy policy categorization and grading.

The second goal is to explore further applications of the machine learning approach towards the privacy policy evaluation. Specifically, we aim to solve the share statement understanding task using text classification, feature engineering and similarity methods. In another word, this part of the work should serve the purpose to demonstrate the potential of applying machine learning to further tasks in privacy policy evaluation beyond the basic job of classifying privacy policy paragraphs into different topics.

Another goal of this work is to propose a generic framework with necessary research facilities that support this thesis project work and related future work. We aim to implement the utility programs, collect the privacy policies, manually label the items and build the datasets for experimental evaluations.

1.3 Thesis outline

The rest of the thesis is constructed as follows:

Chapter 2 proposes an overall framework for the privacy policy evaluation systems that apply the generic machine learning approach. It serves as an overview of the architecture design of the work in this thesis project.

Chapter 3 first defines the categories of privacy policies topics. Then it presents the topic classification system that categories privacy policies into the categories, along with intensive experimental evaluations. Furthermore, it introduces several extensions to the topic classification system.

Chapter 4 focuses on solving a specific task of share statements understanding, using text classification and natural language processing techniques. Two families of methods are presented, implemented and empirically evaluated.

Finally, Chapter 5 presents the conclusions, summarizes the contributions and implications, as well as covers several future research directions.

Chapter 2

Project Design

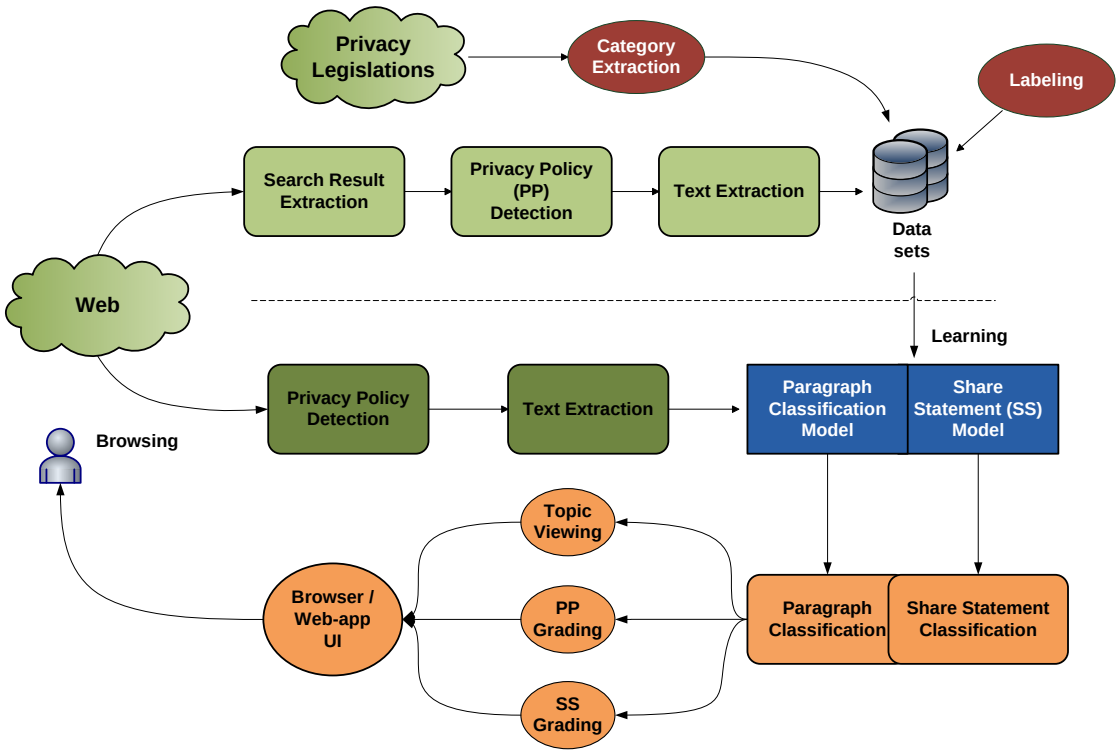
This chapter describes the high-level deign of the work carried out throughout the thesis project. Firstly, we present a general overview of the framework. Then, we focus on several separate modules that are auxiliary components serving for the core components in the framework.

2.1 Design of the framework

We propose an overall framework to apply the generic machine learning approach to automatic privacy policy evaluation. This framework serves as the blueprint for all the work in this thesis project and lays the groundwork for future development as well. Figure 2.1 demonstrates the design of this framework. Its purpose is to help the readers understand how every module contributes to the whole as well as in what ways the procedures interfere one another.

There are three major parts of the whole framework. The first part consists of the light green modules and red modules in the top of the diagram and the datasets. We call this part the **Input interface**. The main function of this part is to receive and process the inputs from outside world and finally form the labeled datasets of privacy policies. The automated modules in light green color create a pipeline in harvesting privacy policies from the Internet. The search result extraction module first grasps the Google search results on keyword ‘privacy policy’ and passes it to the privacy policy detection module. This module detects and selects the privacy policies and further passes them to the text extraction module where the contents of privacy policies are extracted. Besides, there are two manual processes colored in red in the diagram, namely, the category extraction process and the labeling

Figure 2.1: Design of the overall framework



process. Finally, the items of privacy policies are labeled with the pre-defined categories and put into datasets.

The second part of the framework is the **Machine learning core**. It uses the datasets from the first part and applies the machine learning techniques to form the classification models. This part is also the main subject of this thesis and the research work in this thesis project.

The last part of the framework, the **User interface**, handles the interaction between the systems and the users. It is comprised of the dark green and orange colored modules in the diagram. Once a user browses a privacy policy webpage, the detection module sends the page and the extraction module extracts the contents. Then the classification models process this content and generate the results of classification. Based on the classification results, further privacy policy evaluation results, including privacy policy grading, topic-specific viewing and share statement grading, are provided to the user.

To summarize, the framework defines a 3-layer architecture for automatic privacy policy evaluation using machine learning approach. It consists of three major layers – an input interface, a machine learning core and a user interface.

Figure 2.2: High-level conceptual design of the framework

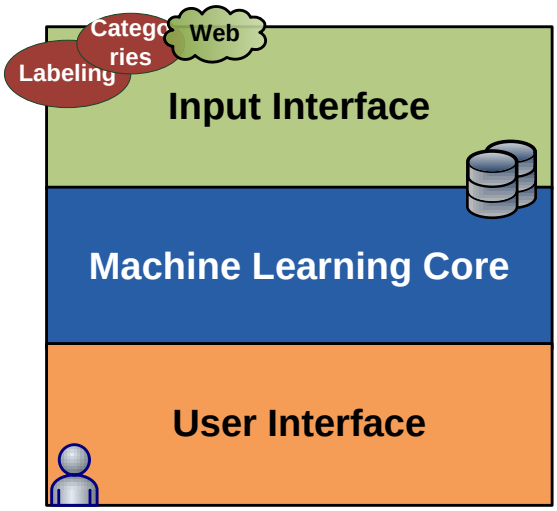


Figure 2.2 is drawn to show this high-level conceptual architecture.

Where are all the components? Because each component in the framework has different importance in our thesis work from a research point of view, this thesis is not structured directly following the three layer architecture. As a report of a piece of scientific research, it mainly focuses on the work of the machine learning core part, yet only briefly covers the work in the other two parts. Here we provide a quick mapping between the components in the framework and the sections in the thesis, for the convenience of the users.

The accessorial components, including all the green modules in the diagram 2.1, are covered in section 2.2. Categories of privacy policy topic classification and categories of share statements are presented in section 3.2 and section 4.1 respectively. Labeling is not covered in this thesis due to its simplicity from both research and implementation viewpoints, however, we do provide the information about the subjectivity test with regarding to the labeling process in section 3.3.1. Privacy policy topic classification datasets are introduced in 3.3.1, the datasets of share statements are covered in 4.4.1 and some the details of datasets are provided in appendix B. Chapter 3 and Chapter 4 together cover the machine learning core part of the framework. Privacy policy grading is presented in 3.4.3 and visualization is covered in 3.4.4.

2.2 Accessorial components

Before we jump into the discussion about topic classification of privacy policy, we will first briefly cover a few functional modules on which the topic classification depends. Namely, we will present the methods used for detecting and extracting of privacy policy contents from web pages. Though may be taken as given, from engineering point of view, these tasks are not entirely trivial and may systematically effect the later steps of classification. Furthermore, we will also introduce the datasets prepared for the experimental evaluations in this section.

2.2.1 Privacy policy detection

A privacy policy detector is an automatic program that distinguishes privacy policies apart from other types of web pages. In an automatic privacy policy evaluation system, the privacy policy detector is the first component in the streamline that picks out privacy policy for further steps.

We choose to implement the detector as a Google Chrome extension, which provides good use-friendliness and flexibility. The core scheme of the Chrome privacy policy detector depends on the assumption that there are many specific topics in privacy policies. The implementation is based on regular expression.

Particularly, when the Google Chrome browser loads a web page, the privacy policy detector, as a Chrome extension, fetches the contents through Chrome API[1] and carries out an efficient run over all the contents so as to match the predefined regular expressions. These predefined regular expressions exploit commonly observed patterns in different topics of the privacy policy. Then the Chrome extension counts and calculates if the contents in the web page hit enough matches of the predefined regular expressions so that to cumulate a matching score higher than a certain threshold. If so, the detector judges the web page as a privacy policy page. The detector then displays a notification bar in the Chrome browser and triggers further processing steps.

Though it is a simple Chrome extension, the detector is very effective with high accuracy, as shown in table 2.1, and low computational cost, due to the application of standard regular expression library for the regex based scoring scheme. For the test set consists of about 900 test cases, which are retrieved from Google

[1] API manual at http://code.google.com/chrome/extensions/api_index.html

Table 2.1: Privacy policy detector - evaluation results on 934 full-text privacy policy dataset

Item	#	In-class(%)	Overall(%)
Whole dataset	934		
Valid items[1]	904		
Non summary items	875		100.00
Privacy policy	796		
True positive	742	93.22	84.80
False negative	54	6.88	6.17
Non privacy policy	79		
True negative	63	79.75	7.20
False positive	16	20.25	1.83
Privacy policy summary	29		
Detected	15	51.72	
Undetected	14	48.28	

1: Exclude non-HTML or unaccessible pages

search results on keyword 'privacy policy', the detector provides up to 92% overall accuracy (combining true positive and true negative). If a web page is a privacy policy, the detector shows 93.22% accuracy in judging it correctly. Moreover, given the fact that all test cases are highly related to privacy issues, the 79.25% accuracy for judging a non-privacy policy correctly is also very reasonable.

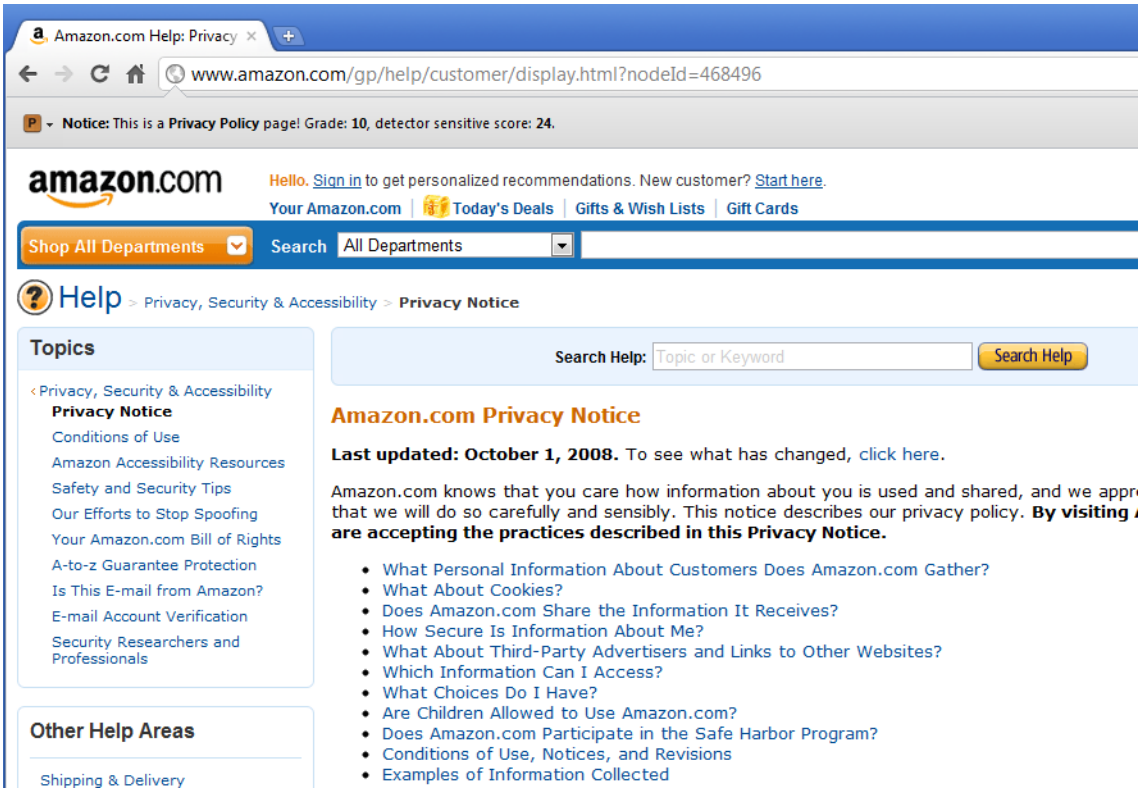
The user interface of this Chrome extension is demonstrated in figure 2.3. When a privacy policy is detected, a notification bar right under the address bar pops up. When the page is not detected as a privacy policy, the extension works quietly in the back-end.

2.2.2 Search result extraction

Gathering the raw inputs, the privacy policies, for our research is a repetitive and labor-intensive task for manual processing. Therefore, we build a module to automatically retrieve the search results from Google. After we define a keyword, e.g. 'privacy policy', all 1000 search results from Google are retrieved as urls. Then the webpages are opened given these urls, and contents are saved. To point out, Google sets a hard limitation and only 1000 results are available on each search keyword. We apply a Python library called xgoogle[2] as the core of this module.

[2] http://www.catonmat.net/blog/python-library-for-google-search/

Figure 2.3: Screenshot of the privacy policy detector



2.2.3 Text extraction

After the detector judges a web page as privacy policy, the next step, before any topic classification occurs, is to extract the main contents of the privacy policy out of the web page. Article text extractor, as its name indicates, is the utility that is capable of distinguishing and extracting the parts of web page which represent an article apart from other common website building blocks like menus, headers, footers, advertisements, etc. Even though it is easy for humans to distinguish the differences, it is still a challenge for a program to detect the main article automatically in the web page.

Different methods have been proposed during the past few years as a respond to the growing demands from web scraping, web/text mining and article reading utilities practice. Kohlschütter et al. [52] propose a text classification based approach that applies shallow text features. Methods using ext-to-tag ratio [106], vision-based page segmentation [16] and maximum subsequence segmentation [75] have been proposed as well.

We apply three libraries in our implementation to extract the main contents

of privacy policies from web pages - Boilerpipe [3], which is a Java library that implements the method described in [52], html2text[4], which is a Python library that converts an HTML page into clean, easy-to-read plain ASCII text, and lxml[5], which is a parser for processing XML and HTML in Python.

[3] http://code.google.com/p/boilerpipe/
[4] http://www.aaronsw.com/2002/html2text/
[5] http://lxml.de/

Chapter 3

Topic Classification

3.1 Overview

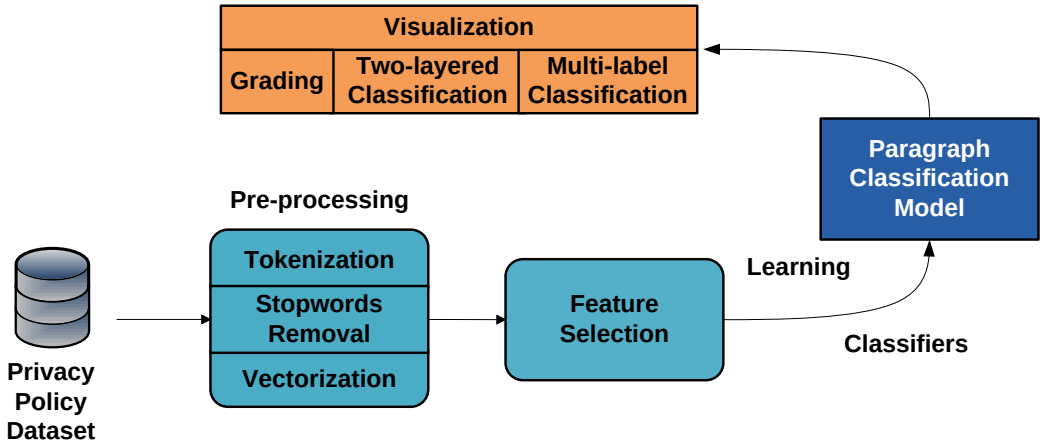
In this chapter, we study the application of machine learning techniques to categorize the privacy policy paragraphs. We call this process 'Topic classification', expressing the nature that privacy policy paragraphs are categorized into specific topics. This privacy policy paragraph classification system is one of the two parts of the machine learning core of our automatic privacy policy evaluation framework.

Besides the major research goal of building the topic classification system, we also aim to cover the extensions of this system, such as privacy policy grading functionality, multi-label and two-layered classification extensions, and visualization of the classification results.

Figure 3.1 demonstrates the overview of the topic classification system. From data perspective, items in the datasets are pre-processed to form the input of the learning phase, where classifiers learn on the training datasets. In the validating phase, classification models are selected. Based on the topic classification models, further extended functionalities are implemented.

The figure is also the general storyline of this chapter. First of all, we cover the definition of categories of topic classification in 3.2 and introduce the related datasets in 3.3.1. Following that, we describe the pre-processing methods of the topic classification in 3.3.2. Then in 3.3.4 and 3.3.5, we briefly introduce the classifiers and ensemble methods, and then test them on the validation dataset in order to choose the best classification models. Finally in 3.3.6, we test the best models on the test dataset. In section 3.4, we present the extensions of the basic topic classification system.

Figure 3.1: Topic classification overview



3.2 Categories

In this section, we define the categories of the topic classification of privacy policy paragraphs. First we present the sources of the categories in a survey manner, and then we list and introduce the actual categories of the topic classification.

3.2.1 Sources of categories

There are two major sources of information for defining the categories – the relevant legislations and the common practice in writing privacy policies.

Legal basis In 1995 the European Union introduced the Data Protection Directive [73], officially Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data, for its member states. The Directive regulates the processing of personal data and sets principles relating to many aspects of the protection of personal data. For example, Section II of the Directive lays down the principles for processing special data, Section IV sets the principle for informing the data subject, Section V defines the data subject's right of access to data, Section VIII is about the confidentiality and security of processing, and Chapter IV - Transfer of personal data to third countries defines limitations on the transfer, processing and storage of the user's information in third countries outside of the EU.

As a preceding effort before the EU 1995 Directive, the Organization for Economic Cooperation and Development (OECD) issued its guidelines on the protection of privacy and trans-border flows of personal data [80] in 1980's so as to create a comprehensive data protection system throughout its member countries. It covers principles for protection of personal data such as notice, purpose, consent, security, disclosure, access and accountability.

As a result of the Directive, many organizations outside of EU began to draft policies to comply with the Directive. U.S. Federal Trade Commission (FTC) published the Fair Information Practice Principles [18] which sets principles for the commercial use of personal information and covers categories such as notice/awareness, choice/consent, access/participation, integrity/security, enforcement/redress and so on.

The Chapter IV - Transfer of personal data to third countries of the EU 1995 Directive requires "the third country in question ensures an adequate level of protection" when personal data is to be transferred outside to countries outside of EU. However, there are differences in the legislative practice between EU and other parts of the world [38, 91], for example, the United States does not have a specific federal regulation establishing universal implementation of privacy policies. Some exceptions to this rule in the Directive are provided. The US-EU Safe Harbor [24], which was developed by the US Department of Commerce, is such a streamlined process for US companies to comply with the Directive if the companies adhere to the 7 principles outlined in the Directive. In particular, these principles are notice, choice, onward transfer, which form the basis for the category '*Sharing*', security, data integrity, access and enforcement.

There are also policies and laws which pose regulations and concerns to the privacy of specific group of people or specific matters. These policies also lead to special practice in drafting the privacy policies. For instance, the Children's Online Privacy Protection Act [19], effective April 21, 2000, applies to the online collection of personal information from children under 13 and forms the basis for the category of '*Children*'. The EU 2006/24/EC Directive [74] deals with the regulation of some important specific privacy-related issues such as confidentiality of information, treatment of traffic data, spam and *cookies*. In the US, some states have implemented strict regulations for privacy policies. An example is the California Online Privacy Protection Act of 2003 ("OPPA") [1].

The paper [41] serves as a comprehensive survey of current regulations on

privacy protection.

Common practice in privacy policies Though not mandating policy, the principles set in the above introduced legislations provide guidance for drafting privacy policies. In order to comply to these policies, there are common practice and even niche market formed. For example, after the Safe Harbor Program worked out by US Department of Commerce, the FTC has approved eTrust[1] to certify streamlined compliance with the US-EU Safe Harbor[2]. Online Certification programs are another example of industry self-regulation of privacy policies. They require implementation of fair information practice as determined by the certification program and may require continued compliance monitoring. TRUSTe[3], eTrust, and Webtrust[4] are some examples of the seal providers.

In practice, there are some other common contents that are normally stated in privacy policies but not explicitly covered by guidelines and principles stated in the legal policies. For example, it is quite common for a website to warn the users about the privacy risk caused by the links to third party websites, to notify the users about third party advertisements and related privacy risks, to state about possible updates of the privacy policy, or to provide contact of the company for inquires or complaints. Another instance is the privacy of location data. With the prevalence of wireless mobile Internet services and location-based social networking services, the privacy issue about location information is gaining more and more attentions and becomes one special category of content in privacy policies.

3.2.2 Core categories

There are four categories that are the main components of privacy policies (see section 3.2.1 for details about the sources of categories):

- Category of **'Collection'** discloses how the company[5] may collect information from the users. Normally, it covers the methods of information collection and descriptions of the information collected. Though 'collection' is not directly

[1] The Electronic Trust foundation (eTrust) is an organization promotes the online privacy through the establishment of best practice and policy. http://www.etrust.org/.

[2] http://en.wikipedia.org/wiki/Privacy_policy

[3] TRUSTe is one of the leading online privacy solution providers. http://www.truste.com/.

[4] http://www.webtrust.org/

[5] We will use the term 'website' and the term 'company' interchangeably when referring to the party who publishes the privacy policy.

addressed by any specific section or article in the directives, it is generally covered throughout these legislations.

- Category of **'Sharing'** states whether the website will share user's information. If so, further statements will normally be provided to answer the questions such as under what conditions will the information be shared and to whom will the information be transferred? The disclosure part of the OECD guidelines is the legal basis of this category.

- Category of **'Choice and Access'** provides very useful information about user's privacy choices, such opt-out options and user's rights to access, amend, modify and/or delete the information collected by the website. Section V of EU Directive 95/46/EC is a direct basis of this category. Choice/consent and access/participation in the FTC's Fair Information Practice Principles are also closely related to this category. Similarly, consent and access in OECD guidelines are also related.

- Category of **'Security'** summarizes the website's practice and standard in protecting user's information. It may refer to the security technologies applied by the website, as well as to the company policies that regulate the employees' practices. Section VIII of EU Directive 95/46/EC is a basis of this category. This category also relates to the security parts of the OECD guidelines and FTC's Fair Information Practice Principles.

3.2.3 Supporting categories

Besides the four core categories introduced above, we also define another 12 categories which act as supporting categories that provide further information about the website's privacy practice. These categories can be sorted into two general types, namely, positive supporting categories and neutral supporting categories.

It is important to point out that the difference between the core categories and supporting categories is solely decided by the amount of content and a category's role in the whole privacy policy. Core categories are the major components of privacy policies and are considered as must-have items in any relatively well-formed privacy policy, while supporting categories are considered to be extra disclosure of information. However, the importance of each category is subject to personal interpretation.

Positive supporting categories Positive supporting categories are the categories that, if appear in a privacy policy, bring positive implications. If a privacy policy contains paragraphs that belong to these positive supporting categories, the privacy policy is considered to be more trustworthy, or at least, this shows the website's positive attitudes in taking user's privacy seriously and providing more detailed information about its privacy practice.

- Category of **'Children'** discloses the company's policy regarding the collection and use of personal information about children. It is closely related to the Children's Online Privacy Protection Act.

- Category of **'TRUSTe'** states the website has been awarded TRUSTe's Privacy Seal signifying that this privacy policy and practice have been reviewed by TRUSTe for compliance with TRUSTe's Privacy Program Requirements available at TRUSTe.com including transparency, accountability and choice regarding the collection and use of user's personal information. This category is defined due to the common practice in the industry.

- Category of **'Safe Harbor'** provides status of a website's participation in and self-compliance with the U.S.-EU/Swiss Safe Harbor Framework as set forth by the U.S. Department of Commerce regarding the collection, use, and retention of data from European Union member countries and Switzerland. Clearly, U.S.-EU/Swiss Safe Harbor Framework is the direct basis for this category.

- Category of **'Link to outside websites'** generally provides notification and warning to users about the hyperlinks to other third party websites which the privacy policy under concern does not cover. This category is defined due to the common practice in the industry.

- Category of **'California Privacy Rights'** states the specific rights for California residents to request information regarding the disclosure of personal information by the website to third parties. The legal basis of this category is the California Online Privacy Protection Act.

Neutral supporting categories In addition to the categories introduced above, there are other supporting categories which simply provide further information

about the website's privacy practice. However, in the occurrence of such categories, whether or not the content is positive for the user is unknown without knowing the details.

- Category of **'Retention'** describes the website's practice in retaining user's personal data. In general, the details may cover the purpose(s), duration and reason(s) of the retention of personal data. Article 15 in Directive 2002/58/EC addresses the issue of retention period of user's personal information.

- Category of **'Processing'** acknowledges the users about the technology aspect of personal information processing. It provides information about where the personal data is transferred to, stored and processed. From legal aspect, it is of particular interest to residents who live outside of the country where data is sotred and processed. Chapter IV of EU Directive 95/46/EC is a basis of this category.

- Category of **'Cookies'** explains the website's use of cookies and other relevant technologies, such as web beacons and flash cookies. It may also state the purpose(s) of applying cookies and types of information stored by the cookies. This category is closely related to the EU 2006/24/EC Directive.

- Category of **'Advertising'** discloses whether the company displays third party advertisements on the website, or its own advertisements administered by third party advertiser. If so, further information about whether and how personal information will be shared to the advertiser may be provided. This category is defined due to the common practice in the industry.

- Category of **'Change'** states how the company will manage the updates of privacy policy, whether and how the users will be informed in case of substantial revisions. This category is defined due to the common practice in the industry.

- Category of **'Location'** explains the specific privacy practice regarding user's location information. This category is defined due to the common practice in the industry.

- Category of **'Contact'** provides company's contact information in case user has further queries with regard to user's privacy. This category is defined due to the common practice in the industry.

3.3 Basic topic classification

In this section, we study the classification of privacy policy, i.e. categorizing contents of privacy policy into the topic categories as defined in previous section. This classification task is the core component of the automatic privacy policy evaluation framework as proposed in this report.

We cover the important aspects of the topic classification, from the pre-processing to the final evaluation on test datasets. And then we introduce its extensions and its application in privacy policy grading.

3.3.1 Datasets

Throughout the discussions about topic classification, we will carry out experimental evaluations to study and compare different methods. To pave the way for clearer evaluations, we now introduce the datasets that are applied in this section.

Training dataset The training set consists of 772 paragraphs extracted from approximately 40 privacy policies, which are all found in the major websites like Google.com, Amazon.com, FoxNews.com, BestBuy.com, etc. Each of these 772 paragraphs is manually labeled into one of the 16 categories.

The labeling process was carried out by one single human judge. In order to prevent systematic inaccuracy caused by subjectivity in labeling, a second human judge has also independently labeled a randomly selected subset of the training set to form a reference for testing the subjectivity.

To be more precise, in the subjectivity test, 102 items are randomly selected out of the 772 training cases. The 102 test item cover all categories expect the categories of '*Link to outside websites*' and '*California Privacy Rights*' which are added after the subjectivity test. The categorical distribution of the test items is made to be as close to the full training dataset as possible during the random selection.

Out of the 102 test items, as many as 93 items (91.18%) are assigned with the same labels by both judges, which means only the labels of 9 items (8.82%) are

disagreed on by the judges. Further, within these 9 disagreed items, 3 of them are either marked as difficult or assigned to the same labels as second choice by the second human judge. As shown by this result, especially by considering the relatively large number of categories in the test, there is a high level of agreement between the two judges on the labels of the training set, which leads to the conclusions that the subjectivity of the human judge is low and is not expected to pose any considerable risk to the correctness of the labels.

Test dataset The test sets contain categories from privacy policy that are graded into four different levels of grade[6]. Our test cases come from 24 privacy policies that are graded into one of the grade groups, namely 4, 6, 8 and 10. The numbers of privacy policies are equally distributed into these four grades, that is to say, 6 privacy policies in each grade group.

Table 3.1: Details about the training and test datasets (By categories)

Category	Training	Test(all)	Test(10)	Test(8)	Test(6)	Test(4)
Collection	121	34	13	10	8	3
Sharing	128	47	17	14	11	5
Choice & Access	107	41	18	13	7	3
Security	50	28	8	9	6	0
Children	33	18	7	7	3	1
TRUSTe	19	5	5	0	0	0
Safe Harbor	23	5	4	1	0	0
Link outs.	32	11	1	6	3	1
California	23	4	2	1	1	0
Retention	13	2	2	0	0	0
Processing	17	8	4	4	0	0
Cookies	85	27	8	10	6	3
Advertising	39	9	4	4	1	0
Change	35	19	6	6	5	2
Location	17	1	0	1	0	0
Contact	30	18	5	6	5	2
Total	772	277	104	92	56	25

Table 3.1 demonstrates the details of the test cases with comparison with the training set as well. Though 6 privacy policies are used for each grade group, it is apparent that privacy policies from the lower grade groups, namely 4 and 6, yield smaller amounts of test items than the privacy policies from the higher grade groups.

[6]The details of grading is introduced in later section3.4.3

Following the common practice in text classification community, we will hold the test datasets as unknown for all classification models until the final round of evaluation based on the test set.

We have 277 test items, consisting of paragraphs that are also sorted into subsets of the test set by grade groups, and 772 training items. In total, we have labeled 1049 sample privacy policy paragraphs to form the datasets for topic classification experiments.

3.3.2 Preprocessing

3.3.2.1 Bag-of-words model

We apply the bag-of-words model to represent the word features for topic classification. Bag-of-words maps the text into a model consisting of unordered collection of words, disregarding grammar and word order. As proved by research in the field of Information Retrieval and the early research in the field of text classification, the bag-of-word is a reasonable simplifying model to represent the text in text classification tasks [47]. Nevertheless, it is important to keep in mind that the bag-of-word representation loses some information from the original text, e.g. the semantic information.

Notation of the bag-of-words We can formally define the notation of the bag-of-words model:

- A *word*, which acts as the basic unit of the whole text collection, is an item from the vocabulary domain $\{1, ..., V\}$. In vector space, each word can be represented as a unit vector with one component equals to one and all the other components equal to zero. Hence, the vth word in the vocabulary is denoted as a unit vector w with $w^v = 1$ and $w^{\bar{v}} = 0$.
- A *document* is represented by a tuple consisting of a list of words $\mathbf{d} = (w_1, w_2, ..., w_{|\mathbf{d}|})$. In our specific task of topic classification, the *paragraphs* retrieved from the online privacy policy are treated as documents in general text classification.
- A *corpus* as a collection of documents is denoted by $\mathbb{D} = \{\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_{|\mathbb{D}|}\}$, where $\mathbf{d}_n$ is the nth document in the corpus. The training dataset and the test dataset are corpora.

- A *category*, also called *class*, is human defined *label* attached to documents. Classes are used, in specific application, to categorize the documents in the document domain into subsets. We define the domain of classes as $\mathbb{C} = \{c_1, c_2, ..., c_{|\mathbb{C}|}\}$.

A *dataset* $\mathbb{D}$ of labeled documents is therefore denoted by $\langle d, c\rangle$ where $\langle d, c\rangle \in \mathbb{D} \times \mathbb{C}$.

3.3.2.2 Tf-idf

Based on the bag-of-words model, the term frequency-inverse document frequency (tf-idf) weighting [88], which is a widely used vector space weighting scheme in information retrieval and text mining tasks, is applied on each value in the bag-of-words representation instead of the simple word frequency. In contrast to the pure word frequency, tf-idf weighting has advantage, which is brought by the inverse document frequency factor that weights high frequency term less and rare term more. This is beneficial because, in text classification, the rare terms are considered to be more valuable in distinguishing categories for a given test item. Other weighting methods have been proposed, such as in [99, 32], however, tf-idf is still one of the most commonly accepted methods for preprocessing the vector representation for text classification tasks [95].

3.3.2.3 Feature selection

In text classification, a dimensionality reduction process is often carried out in order to reduce the size of the bag-of-words representation from $|\mathbf{d}|$ to a smaller and normally predefined number. There are two major reasons for dimensionality reduction: 1) to reduce overfitting, i.e. the model should have low *variance* and should not overfit on the training data and lose the ability to generalize to unseen data; 2) to alleviate the *'curse of dimensionality'* [39, 7] for the learning methods that are known to scale badly to high dimensionality.

Feature selection is normally used for dimensionality reduction in text classification. During feature selection, each word in the bag-of-words model is scored by a scoring function that captures its degree of correlation with category c_i. Then only the words with highest scores are selected for the final document representation. In our experiments, we use predefined thresholds for feature selection, meaning the numbers of selected words are predefined for each specific experiment.

Feature selection is an active research fields in text classification and machine learning communities. Many methods have been proposed for text classification with focus on certain learning methods. [31, 114] provide systematic survey on this issue. In later experiments, we will apply the χ^2 feature selection method.

3.3.3 Evaluation of text classification

For text classification, throughout this report, we will use the generic term 'performance' when measuring the quality of classification decisions. Though, the term 'performance' is also commonly used to express computational efficiency of classification [58], we only use it for effectiveness of classification unless otherwise indicated.

3.3.3.1 Metrics

For text classification, the terms *true positives (TP)*, *true negatives (TN)*, *false positives (FP)*, and *false negatives (FN)* compare classifier's outputs on the test set with the predefined labels which are created by external judgments such as human annotators. Positive and negative refer to the prediction generated by the classifier, and true and false refer to whether a prediction corresponds to the predefined label.

The metrics commonly reported for text classification are defined as below:

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$Error = \frac{FP + FN}{TP + TN + FP + FN}$$

F_β measures are used to provide a balanced view of the performance by weighting and combining precision and recall. Most commonly used form of F_β is F_1 which is the harmonic mean of precision and recall. Two other commonly used F_β measures are the F_2 measure, which weights recall higher than precision, and the $F0.5$ measure, which emphasizes more on precision than recall.

Selection of metrics for text classification

- **Accuracy**, especially in multi-classification tasks where the number of classes is relatively high and the distribution of samples among classes are relatively flat, is not a reliable metric for text classification. This is due to the high imbalance between the amounts of positive examples and the amounts of negative examples for multi-class tasks. This means there are too many *tn* that dominate the result of accuracy. This can lead to miss-interpretation of the results conveyed by accuracy.

  For instance, when the positive examples of a category constitute only 10% of the entire test set, a dummy classifier that makes negative predictions for all documents has an accuracy of 90%, or an error of only 10%. However, such a system is meaningless in most of the cases. For this fact, *we are more interested in using F scores, precision and recall instead of accuracy and error to evaluate our tasks.*

- **F**$_{0.5}$: Intuitively, precision shows a classifier's ability not to label negative samples as positive, and recall shows a classifier's ability to label as many positive samples as possible. F_1 is the harmonic mean of precision and recall, which is not biased on either precision or recall. For the reason that precision is more important than recall from user's prospective, we will also measure the $F_{0.5}$, which is a variant of F_β that is biased on precision.

Averaging methods In case of the binary text classification tasks, the metrics mentioned above are calculated simply using TP, TN, FP and FN. However, for multi-class text classification, averaging scheme is needed to calculate the overall metrics from the TP, TN, FP and FN. For this reason, we will introduce three averaging schemes. First, we will use the terms TP_i, TN_i, FP_i and FN_i to denote the basic metrics of category i. Three averaging schemes are shown as below:

The Micro-averaging leads to precision and recall that are directly averaged globally, meaning that the micro-averaged metrics give an equal weight to each document.

$$Precision_{micro} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)}$$
$$Recall_{micro} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FN_i)}$$

In contrast, the macro-averaging generates precision and recall on the category level. The macro-averaged metrics are calculated based on simple average of the category metrics. This means that equal weight are given to categories rather than documents.

$$Precision_{macro} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{TP_i}{TP_i + FP_i}$$
$$Recall_{macro} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{TP_i}{TP_i + FN_i}$$

These two different averaging schemes are distinctive especially when categories are unbalanced and have varied performances. Macro-averaging favors more on the rare categories because all categories are considered equally. Macro-averaging is informative particularly when the performances of rare categories are more concerned. Micro-averaging, on the contrary, reflects the performance on the level of all documents and therefore is inclined to the major categories.

The problem with macro-averaging is that it only conveys the performance on the level of category and it is often dominated by the rare categories. When the categorical distribution of test dataset reflects the distribution of future real dataset, macro-averaging may be misleading by favoring the models that perform better on rare categories which are much less common than few major categories. This could bring forth the risk of a biased model selection, so that the selected models has much worse overall performance in practice.

In general tasks of text classification, micro-averaging is more commonly used [94, 34]. However, for micro-averaging, in case mutually exclusive categories are

used, the precision and recall will always be the same [14]. This can be easily shown as the $\sum_{i=1}^{|C|} FP_i$ always equals to $\sum_{i=1}^{|C|} FN_i$ when categories are mutually exclusive. If the $Precision$, $Recall$ and F_{scores} are the same, the information about the differences between $Precision$ and $Recall$ is missed out.

For the need of our specific task, both micro-averaging and macro-averaging have their own limitations as stated above. We apply a third averaging scheme, the weighted averaging, as described below. It is generally a micro-averaging scheme in that the final metrics are averaged to the global number of documents. But it also shares similarity with the macro-averaging in that the basic metrics, from which averaged scores are computed, are calculated on the category level.

$$Precision_{weighted} = \frac{\sum_{i=1}^{|C|} |D_{C_i}| \frac{TP_i}{TP_i+FP_i}}{|D|}$$
$$Recall_{weighted} = \frac{\sum_{i=1}^{|C|} |D_{C_i}| \frac{TP_i}{TP_i+FN_i}}{|D|}$$

where $|D|$ denotes the total number of documents and $|D_{C_i}|$ denotes the number of documents for category C_i.

This averaging scheme generates scores that are very close to the micro-averaging while still preserves the differences between $Precision$, $Recall$ and F_{scores}. It also partially preserves the category metrics as in macro-averaging but will not be dominated by the rare categories. Due to the reasons above, we will apply the weighted metrics unless otherwise mentioned.

3.3.3.2 Cross validation

To largely utilize the limited resources in a statistically accurate way, we heavily apply cross-validation throughout our experiments. The usages of cross-validation are [83]: 1) To gauge the generalization of a learned model based on the limited amount of available data. 2) To compare the performance of different algorithms for a specific learning task. 3) To compare different models, including comparisons of parameter tunned models of parameterized classifiers, and comparisons of different feature engineering techniques.

There are different variants of cross-validation. As suggested by Kohavi [51], *stratified 10-fold cross-validation* is preferred over leave-one-out cross-validation

and bootstrap. Due to the limited samples (< 10) in some rare categories in our test sets, we cannot carried out stratified 10-fold cross-validation in all scenarios. Hence, to form a consistent evaluation baseline, we will follow the widely accepted practice in machine learning community by applying the *10-fold cross-validation*.

Further, if statistical accuracy is more demanding for certain situations when we compare two target models, $n \times 10$ cross-validation (n could be 10, 50, etc) will be applied to provide more samples. The approach of 10×10 fold cross-validation is also recommended by Bouckaert [8]. To compare two methods more precisely, Salzberg [89] suggested using k-fold cross-validation followed by appropriate hypothesis test rather than directly comparing the applied metrics. To this end, we also apply *paired t-test* which is widely used in machine learning studies.

3.3.4 Classification model

With the categories of text classification and preprocessing steps introduced, we now address the selection of optimized classification model for our specific task - the topic classification of privacy policy paragraphs. Classification model, also known as classifier or learning method, is the core of a text classification task. In this section, we will first introduce a basic formal model of classification, and then cover several classifiers that are commonly used in text classification. Further, experiments will be carried out to tune the classification models and compare the classifiers. Last but not the least, we will also apply the ensemble learning methods in our task.

3.3.4.1 Definition of classification

Suppose we have a training set with n data items. Let the data items in the training dataset be vectors denoted by $\mathbf{x} = (x_1, \ldots, x_n)^T$, which are actually the preprocessed paragraphs in our specific task of text classification. For the training set, we already have the target variables, a.k.a. categories or labels, denoted as $\mathbf{y} = (y_1, \ldots, y_n)^T$. In the *learning* stage (also called *inference* stage), the aim is to learn a set of parameters θ of the classification model, which infers a classification model with a hypothesis classification function $h_\theta(\cdot)$, based on the given $\mathbf{x}$ and $\mathbf{y}$.

With this model learned, later in the *prediction* stage (also called *decision* stage), given we have a test set with m test data items which is denoted as vector $\mathbf{x}_{\text{test}} = (x_1, \ldots, x_m)^T$, we want to predict the targets of the test set, denoted

as $\mathbf{y}_{\text{test}} = (y_1, \ldots, y_m)^T$, using the classification model. This process is done by applying the method learned, which is written as $\mathbf{y}_{\text{test}} = h_\theta(\mathbf{x}_{\text{test}})$.

Another common notation of classification is formed by using $\mathbf{w}$ denoting the parameters. In this notation, the prediction is written as $y(\mathbf{x}, \mathbf{w})$. Later, we may use the two slightly different notations interchangeably. However, it is necessary to point out $\mathbf{w}$ here should be confused with the representation of documents in the bag-of-words discussion.

3.3.4.2 Linear classifiers

A generalized linear classification model is denoted as $\mathbf{y} = \theta^T\mathbf{x}$, where $\theta^T = (\theta_0, \ldots, \theta_p)$ with p the number of datapoints and θ_0 the bias. Compared with non-linear models such as k-nearest neighbors, linear models depend more on the assumptions about problem's structure and yield more stable predictions. Such linear models have been a mainstay of statistics for past decades and widely studied and applied in statistical learning and machine learning communities. We will briefly cover three linear classifiers, namely, Naive Bayes, Ridge regression and linear SVM.

Ridge regression Given the linear model $\mathbf{y} = \theta^T\mathbf{x}$, there are many methods to fit the model to a set of training data. The method of *least squares* is one of the most popular methods for this purpose. In the simplest form, the least square can be denoted as $\frac{1}{2}\|\mathbf{w}^T\mathbf{x} - \mathbf{y}\|_2{}^2$. So, one can fit the model to the training data by minimizing the least squares, i.e. $\min_w \frac{1}{2}\|\mathbf{w}^T\mathbf{x} - \mathbf{y}\|_2{}^2$.

The solution for minimizing the least squares above is given by $\mathbf{w} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$. However, there is a problem with this simple form of least square, i.e. $\mathbf{x}^T\mathbf{x}$ may be singular or ill-conditioned, e.g. when the number of training data n is smaller than the dimension of $\mathbf{x}$. In this case, a unique solution cannot be guaranteed.

Ridge regression [42] classifier [119] is one of the linear classification models that remedy this problem by adding a regularization term to the standard least squares. The Ridge regression variant of least squares and its solution are then written as

$$\min_{w} \frac{1}{2}\|\mathbf{w}^T\mathbf{x} - \mathbf{y}\|_2{}^2 + \lambda\|\mathbf{w}\|_2{}^2$$
$$\mathbf{w} = (\mathbf{x}^T\mathbf{x} + \lambda n\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y}$$

where **I** is the identity matrix. By adding the Ridge regularization term to the least squares, the $\mathbf{x}^T\mathbf{x} + \lambda n\mathbf{I}$ part of solution is always non-singular, provided that $\lambda > 0$. This solves the singularity problem.

Another merit of Ridge regression classifier is its 'cost effectiveness'. It is more computational efficient compared with many other commonly used linear regularized methods such as linear SVM and regularized logistic regression. Ridge regression classifier performs on the same level or even slightly better than the other two methods [118].

Linear SVM SVM has been widely and successfully applied for text classification since the late 1990's [48, 23]. We will cover the linear SVM here, which is one of the most popular kernels for text classification [118]. Later we will also briefly introduce another nonlinear kernel based SVM.

The linear SVM projects datapoints to a higher dimensional space and tries to find the optimal hyperplane with maximized margin, which has the largest distance to the nearest training datapoints. Assume the training datapoints are linearly separable in the higher dimensional space, a method, which applies two auxiliary hyperplanes, can be used to find the searching hyperplane. In this method, the two auxiliary margin hyperplanes are searched in a way that there should be no datapoint between these two auxiliary margin hyperplanes and the searching hyperplane while trying to maximize their distances to the searching hyperplane. This is equivalent to minimizing norm of the weight vector:

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{subject to: } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i$$

Corinna Cortes and Vladimir Vapnik proposed a more advanced maximum margin method, called the Soft Margin [20], that allows for the mislabeled items:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to: } \xi_i \geq 1, \forall i$$
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \forall i$$

where C represents cost coefficient and ξ_i is slack variable that measures how far the corresponding data falls into the wrong side of the margin.

The explanation above forms a very brief introduction to the SVM by covering the formal representations of the intuitions behind the hyperplane searching. We recommend further readings [15, 105] for a thorough introduction.

Generative models The generalized linear classification model described above is also a *discriminative model*. In contrast to the discriminative models, such as Ridge and SVM, there is another widely used classification model – the generative model. A generative model is a full probabilistic model of both input and target variables, whereas a discriminative model provides a model only for the target variables conditional on the input variables. That is, generative models allow to generate samples from the joint distribution of input and target variables.

Naive Bayes Naive Bayes is one of the most commonly used generative learning methods and has been widely applied to text classification tasks [60, 63, 90]. In simple terms, Naive Bayes is a group of learning methods that all depend on the Bayes' theorem ($p(c|f_1, \ldots, f_n) \propto p(c)p(f_1, \ldots, f_n|c)$, where c is a class variable and $f_1, \ldots, f_n$ are features) along with the 'naive' assumption of the independence relation between every pair of features, i.e. given the target variable, the presence of a particular feature of a class is unrelated to the presence of any other features.

With this 'naive' assumption, the relation based on Bayes' theorem becomes $p(c|f_1, \ldots, f_n) \propto p(c)\prod_{i=1}^{n} p(f_i|c)$. Therefore, model can find the best class by

$$c = \arg\max_{c} p(c)\prod_{i=1}^{n} p(f_i|c)$$

We can use maximum a posteriori probability (MAP) estimate to estimate $p(c)$ and $p(f_i|c)$.

It is necessary to point out again that Naive Bayes is not a single learning method. Rather, the name 'Naive Bayes' represents a group of close related methods applying both the Bayes' theorem and the 'naive' assumption. There are several variants of Naive Bayes, with the difference in the assumptions of the distribution of $\propto p(c)\prod_{i=1}^{n} p(f_i|c)$.

There are two classic variants of Naive Bayes commonly used for text classification, namely the **Multinomial Naive Bayes** and **Multi-variate Bernoulli Naive Bayes** (or simply Bernoulli Naive Bayes). Bernoulli Naive Bayes assumes data to be multi-variate Bernoulli distributions, i.e. each feature is only assumed to be a binary-valued (or say Bernoulli, boolean) variable. The Multinomial Naive Bayes uses multinomial distribution which is parameterized with smoothed maximum likelihood calculated from the distribution information learned from the training set. McCallum et al. in [60] provide a coherent explanation of the differences between the two variants and their applications in text classification.

We will apply the multinomial variant of Naive Bayes in our topic classification. Because the distribution assumption made by multinomial Naive Bayes, it is considered to be more suitable for general text classification tasks, and it indeed performs better than the Bernoulli Naive Bayes in many cases [60].

Naive Bayes classifiers can be much faster than some of the more sophisticated learning methods. The 'naive' assumption leads to the decoupling of the class conditional feature distributions. Therefore it means that each distribution can be independently estimated and calculated as a one dimensional distribution. This also helps to alleviate the *curse of dimensionality* problem. For these merits, Naive Bayes has been widely applied in many text classification tasks, especially in practical tasks. For example, it is widely applied in both commercial and open source spam filtering solutions [63].

3.3.4.3 Non-linear classifiers

'Non-linear classifier' is not a strictly defined term. In general, non-linear classifiers differ from the linear classifiers by the fact that they achieve the classification decision based on non-linear combination of the features. We will cover three classic non-linear classifiers, namely k-nearest neighbor (kNN), decision tree, and SVM with non-linear kernel.

kNN The idea behind kNN is simple – classifying test datapoints based on k closest training examples in the feature space. This idea also leads to its special learning scheme, called lazy learning. In a lazy learning scheme, function is only approximated locally (with k nearest neighbor) and all computations are deferred until the classification of the test datapoints. The nonlinearity of kNN can be easily observed from classification examples [39, 7].

Tree Decision tree is a non-parametric learning method that uses a decision tree algorithm as a classification model. The decision tree algorithm first learns the simple decision rules inferred from data features in the training set. Then observations about a test datapoint are mapped to conclusions about the datapoint's target value by applying the learned decision rules.

An outstanding merit of the decision tree classifier is that it is a white box model. The learned model and results are easily understandable and interpretable. There are also many problems with the decision tree classifier. Firstly, the learned rules may be over-complex and do not generalize well. Secondly, decision trees may be biased if there exists imbalance in categories. Last but not the least, high variance may exist even when a small change occurs in the training data [39].

There are several classic decision tree algorithms, such as ID3 [81], C4.5 [82] and CART [9]. We apply CART algorithm in the later experiments.

SVM with non-linear kernel There are several commonly used alternatives [98, 100] to the linear kernel used in the linear SVM. Popular non-linear SVM kernels are radial basis function (RBF) networks, polynomials, sigmoid, splines and so on.

The linear kernel is denoted as $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$ and the RBF kernel is written as $K(\mathbf{x}, \mathbf{y}) = exp\{-\|\mathbf{x} - \mathbf{y}\|_2{}^2/\gamma^2\}$.

Several common kernels are thoroughly compared from both theoretical and practical aspects in [101]. It is shown that linear kernel is a special case of the RBF kernel. For general tasks, RBF kernel is recommended as the first choice for practical uses. Therefore, we will use the radial basis function kernel for the non-linear version of SVM in later experiments.

3.3.4.4 Parameter tuning

Tuning a classifier involves tuning the thresholds and alterable parameters, depending on the classifier applied. Tuning a single parameter p is normally done

experimentally by performing repeated tests with different values of the parameter p on the validation set, or using cross validation, while fixing the values of all other parameters p_k. Finally, the value that yields the best performance is the result for the tuning of p.

We apply the grid search method for parameter tuning. Grid search performs an exhaustive search through a predefined subset of the parameter space to solve the problem of model parameter selection by finding the optimal parameters guided by certain performance metric under cross validation on the validation set.

SVM comes with several parameters, which differ depending on the applied variant and implementation of SVM. This requires elaborated efforts in tuning the parameters to achieve reasonable performance close to the full potential of the classifier. For SVM with RBF kernel, we tune parameter C and γ following the guide in [43], i.e. using a loose search followed by a fine search.

For *linear SVM*, two parameters are alterable – the regularization method, either l1-norm or l2-norm, and the parameter C. For regularization, as stated in [69], l1 is expected to be theoretically superior to l2 if the number of features is considerably bigger than number of examples and the ground truth, or say asymptotic set of predictive features, must be sparse in the basis. However, when it comes down to specific task in real life, it still depends on questions such as what are the reasons or unwanted behaviors that require regularization and which type of regularization suits for the practical purpose.

For *kNN*, the only alterable parameter is k, which decides how many neighbors are used to make the decision. For *decision tree*, max depth of the tree and min split at a leaf node are the parameters to control the building process of the decision tree.

Ridge regression and *Naive Bayes* are not subjects for parameter tuning. This simplicity becomes one of the advantages of these two classification models.

The parameter tuning results can be found in Appendix C.1.

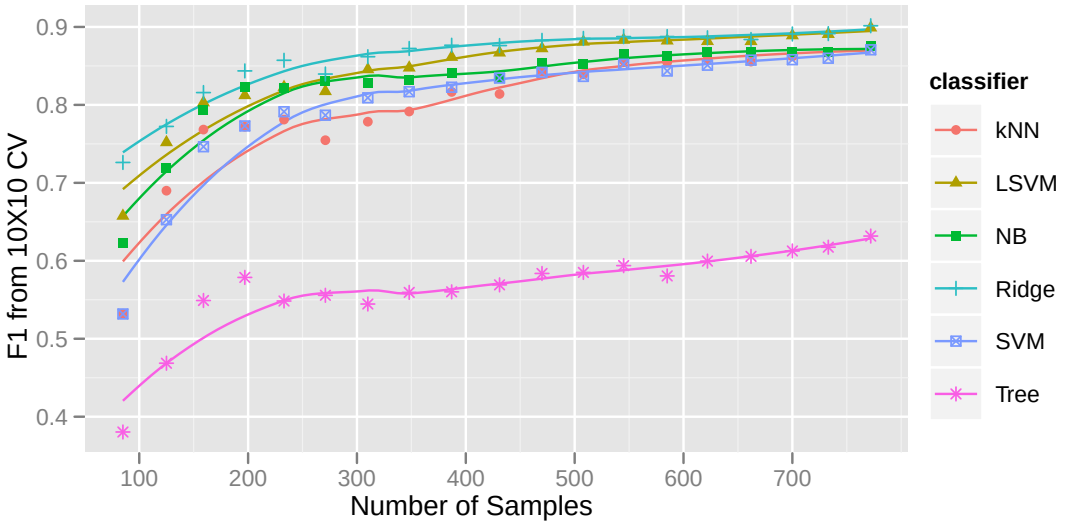
3.3.4.5 Comparison of classifiers

In this section, we compare the performance of the six classifiers in order to gain an overall understanding of the differences among these classifiers and to pave the way for the final classification model selection.

First, we study the performance of the classifiers when the size of training set increases. To this end, we create 18 subsets of the training set by increasing the

percentage of selected items by 5% each time, starting from 10% of the whole training set. The six classifiers are tested separately on these 18 subsets along with the full size training set.

Figure 3.2: Comparison of classifiers (by size of training set) - I



Among all classifiers in figure 3.2[7], the Ridge regression classifier shows superiority over all percentages of the training set. The performance of linear SVM becomes very close to Ridge regression's performance when 70% or more of the training set is used. On the other side, the decision tree classifier shows worse performance than all the other classifiers. This is probably because of the limited capability of a single decision tree for text classification task where the representation is sparse and the number of features is large.

As it can be observed in the results, the differences in classifiers' performance are rooted in the differences among different types of classifiers. Specifically, the linear discriminative models (Ridge regression and linear SVM) perform better than linear generative model (Naive Bayes) and nonlinear models(SVM with RBF kernel[8], kNN[9] and decision tree).

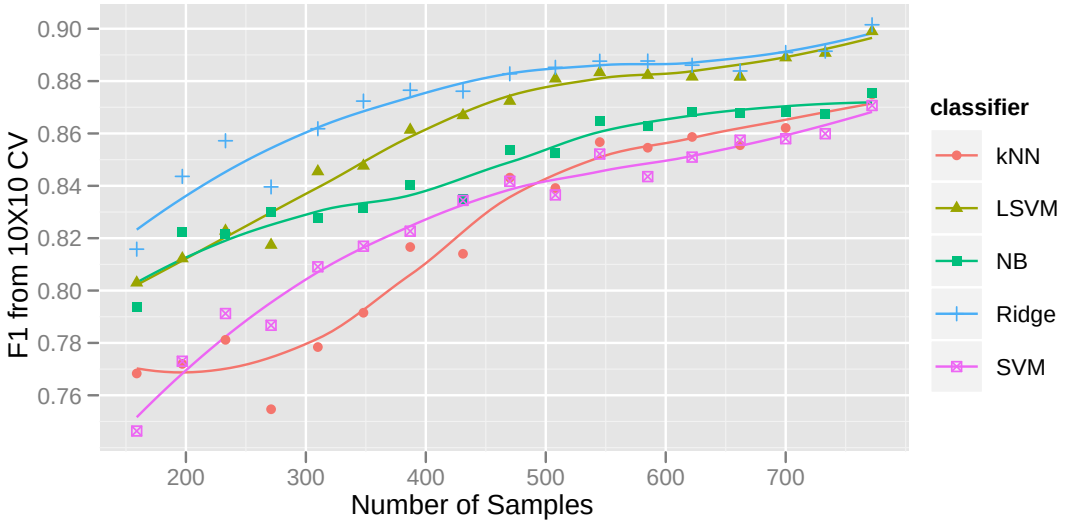
Figure 3.3 provides a zoom in view of the figure 3.2 where the decision tree and the first two datapoints on the x-axis are removed. Given the obvious inferiority of

[7]LSVM in the figure stands for Linear SVM

[8]In this experiment, we set $\gamma = 1/\#\text{feature}$ instead of a constant number because the number of samples are changing.

[9]Similarly, we use $k = \frac{\#\text{feature}}{2*\#\text{class}}$ for kNN.

Figure 3.3: Comparison of classifiers (by size of training set) - II



the single decision tree, we will exclude it in the later experiments unless needed for special purpose.

Second, we examine the performance of the classifiers under χ^2 feature selection. We test all levels of aggressiveness, ranging from 2% to 100% with 2% steps.

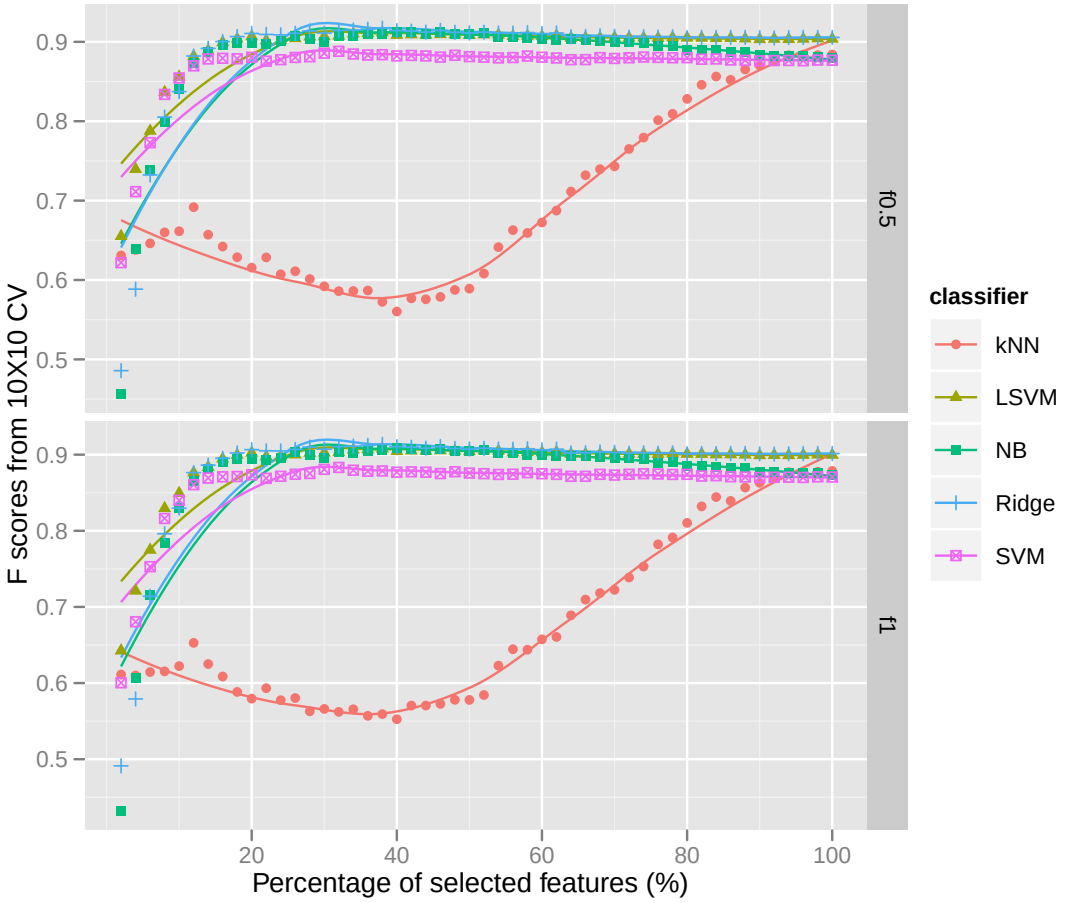
The result is shown in figure 3.4. It shows similar rankings of the classifiers as in previous experiment. This view also reveals kNN's different behavior compared with other four classifiers. Two linear discriminative models still perform very well with slight dropping after the peak. Naive Bayes reaches its peak at similar positions as the linear discriminative classifiers with competitive performance. However, Naive Bayes' dropping at the tail part is much sharper. And fianlly it reaches the level of performance similar to nonlinear SVM and kNN at the end of the tail part.

From this point of view, it is clear that if feature selection method will be used, the two discriminative classifiers still are considered to be better choices.

Among the other three classifiers, though they reach the same level when most of the features are selected, Naive Bayes provides the best performance while kNN generates the worst performance when fewer features are selected.

Commonly, a linear classifier is applied when computational efficiency of classification is crucial, since it is often faster than non-linear classifiers (kNN is an exception), especially when $\mathbf{x}$ is sparse. Moreover, linear classifiers often work quite well when the number of dimensions in $\mathbf{x}$ is large. These features suits very well with the text classification tasks where $\mathbf{x}$ is sparse in the bag-of-words

Figure 3.4: Comparison of classifiers (by number of selected features)



representations.

Though theoretically similar to each other, Ridge classifier, as a regularized least square (RLS) classifier, performs slightly better than linear SVM. Moreover, Ridge is slightly more computational efficient than linear SVM. Similar results have been observed in [86] for text classification on the 20NewsGroup corpus[10]. For more details, Zhang et al. [118] provide a thorough comparison among Ridge, linear SVM and logistic regression under the setting of text classification.

In conclusion, combining the empirical results with theoretical features of different classifiers, our topic classification task is linear separable. This also explains the top-rate performance of the two discriminative classifiers. However, there is no strong evidence that Ridge regression's slight superiority over linear SVM

[10]A popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. http://people.csail.mit.edu/jrennie/20Newsgroups/

will be stable and generalizable. Naive Bayes performs the best among the second tier classifiers and has strong advantage due to its computational efficiency. In contrast to linear SVM, nonlinear SVM shows extra computational cost and slightly worse performance. kNN, though with acceptable performance, is considered to be inferior because of its behavior when feature selection is applied.

3.3.5 Ensemble learning

The purpose of applying ensemble learning methods is to strategically generate and combine classifiers in order to solve a particular machine learning problem. In classification, ensemble learning methods are primarily applied to improve the performance of a classification model by forming an ensemble of multiple classifiers.

There are two commonly used ensemble methods: 1) *Bagging* [10], which stands for *bootstrap aggregating*, generates a group of similar classifiers from one certain type of classifier by using bootstrapped replicas of the training data. 2) *Boosting* creates an ensemble of classifiers by re-sampling the data. These classifiers are then combined by majority voting, in a manner that emphasizes the training instances previously misclassified by preceding classifiers. *AdaBoost* [33], stands for adaptive boosting, is the most popular boosting method so far and generalizes boosting to multi-class and regression tasks.

However, both bagging and boosting are generally applied to one single type of classifier by creating the ensemble using duplications of the same type classifier. We want to create ensembles from totally different types of classifiers, such as SVM, Ridge regression and Naive Bayes. To this end, we implement and test two types of ensemble methods, viz., voting committee and stack generalization.

Voting committee combines the classifiers of different types into a voting committee. On each test datapoint, the classifiers in the committee output the prediction labels based on which a majority vote is cast so as to form the final committee prediction label [12].

Stack generalization [110], also known as stacking or blending, introduces a layered architect of classifiers. In the first tier, classifiers either in different types or created from a single type of classifiers using bootstrapped samples are treated similarly as in the other types of ensemble methods. In the second tier, a tier 2 classifier, or meta-classifier, is applied on top of the outputs from the tier 1 classifiers. In other words, in the layered structure, the final prediction is generated from the

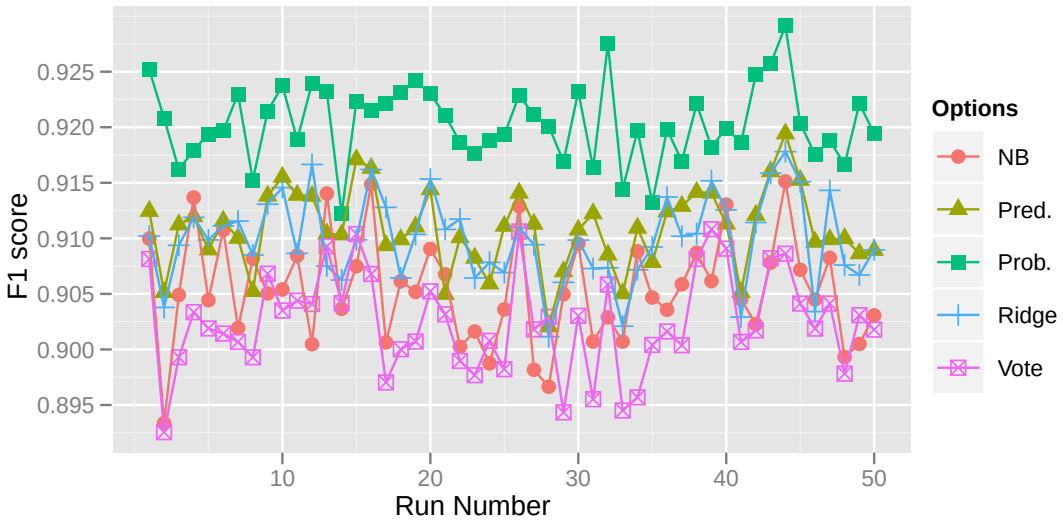
tier 2 classifier which learns from the outputs from the tier 1 classifiers.

Stacking method is gaining more popularity, especially in practical machine learning challenges. For instance, it has been heavily applied in the solutions proposed by the top two teams during the Netflix Prize[11] [6, 77]. In research, it also shows good results in classification and information extraction tasks [102, 97].

For experimental evaluation purpose, we implement and test two variants of the stacking method. Specifically, the first variant implements the tier 2 classifier in the way that it learns from the predicted category labels generated by the tier 1 classifiers. In the second variant of stacking, the tier 2 classifier learns from a linear combinations of prediction scores from the tier 1 classifiers.

In figure 3.5 we show the F_1 performance of the three ensemble methods[12] in contrast to two single learning methods, Ridge and Naive Bayes.

Figure 3.5: Ensemble methods



The majority voting method provides slightly worse F score performance than Ridge and Naive Bayes. This can be explained by kNN's poor performance when about 40% features are selected, which is the setting for this experiment.

On the other hand, the stacking ensemble learning method improves the performance. As can be observed from both figure 3.5 and table 3.2, the prediction

[11] $1M Grand Prize for best the collaborative filtering algorithm to predict user ratings for films based on previous ratings. http://www.netflixprize.com/

[12] 'Pred.' and 'prob.' represent the two variants of stack generalization, i.e. stacking on the predictions and stacking on the categorical probability scores. 'Vote' stands for the voting committee ensemble method.

variant of stacking only provides a small improvement, while the probability score variant of stacking improves the F score performance by more than 1% compared with the best single classifier, Ridge regression.

Table 3.2: Ensemble methods

Category	${F_{0.5}}^*$	${F_1}^*$	Paired t-test
Voting	90.340 ± 0.220	90.249 ± 0.223	
Stacking (Pred.)	91.439 ± 0.189	91.085 ± 0.178	No
Stacking (Prob.)	92.375 ± 0.168	92.036 ± 0.174	Yes
Naive Bayes	90.907 ± 0.258	90.537 ± 0.242	
Ridge reg.	91.362 ± 0.209	90.989 ± 0.195	Baseline
Linear SVM	90.710 ± 0.206	90.325 ± 0.191	
kNN	55.670 ± 0.583	55.227 ± 0.452	
SVM (RBF)	88.103 ± 0.243	87.609 ± 0.232	

*: Averaged from 50 runs of 10×10 CV, in percentage.

In table 3.2, the detailed F scores are provided for all single classifiers and three ensemble learning methods. We also apply paired t-test to check whether the two variants of stacking methods significantly improve the performance compared to Ridge regression, the best single classifier. The other pairs of classifiers and ensemble methods have not been paired t-tested because the differences are obviously distinguishable.

Another observation is that the two stacking methods also reduce the standard deviation, which leads to better generalization capability.

To conclude, stacking methods can improve the performance to at least the same level as the best single classifier while stacking methods are expected to provide better generalization capability for testing datapoints. In the training experiments, the probability score variant of stacking provides statistical significant F score improvement compared with the best single classifier. Voting method depends more on the average level of the classifiers and cannot therefore guarantee to provide top-level performance for all cases.

However, the improvements in both F scores and generalization capability created by stacking methods are limited by a trade-off between performance and computational efficiency.

To be more specific, majority voting's computational cost is the sum of computational costs of all underlying classifiers. Analyzing computational cost of stacking is less straightforward. In training phase, we apply a 10-fold CV process to generates the outputs of each classifier. Based on these outputs, the tier 2 classifier learns.

Hence, the stacking's trainig cost is approximately 10 times full round costs of all underlying tier 1 classifiers. This leads to the conclusion that the selection of models still depends on the practical requirements for computational efficiency and performance.

In general, Naive Bayes is a better choice when both training time and test time are crucial. Ridge regression and linear SVM provides top ranked performance but with training computational cost up to 10 to 20 times higher than that of Naive Bayes. Stacking method is preferred when training computational cost is unimportant while stable top-level performance is of greater interests.

3.3.6 Final test

We test the classification models on the test dataset and its four subsets.

Table 3.3: Evaluation results on test dataset

Dataset	Test(all)		Test(10)		Test(8)		Test(6)		Test(4)	
Metric	$F_{0.5}$	F_1	$F_{0.5}$	F_1	$F_{0.5}$	F_1	$F_{0.5}$	F_1	$F_{0.5}$	F_1
Voting	93.61	93.57	90.27	89.95	97.97	97.91	94.82	94.75	92.53	92.33
Stack.Pred.	93.61	93.57	90.27	89.95	97.97	97.91	94.82	94.75	92.53	92.33
Stack.Prob.	92.63	92.55	88.13	87.52	97.97	97.91	94.82	94.75	92.53	92.33
Naive Bayes	89.86	89.74	89.66	89.21	92.38	92.38	89.54	88.76	90.38	89.47
Ridge reg.	93.73	93.64	90.27	89.95	98.26	98.10	94.82	94.75	92.53	92.33
Linear SVM	93.95	93.92	90.07	89.82	98.98	98.95	94.82	94.75	92.53	92.33
kNN	90.63	90.35	87.63	86.85	95.33	95.04	92.46	91.93	89.06	88.66
Tree	68.96	67.27	67.72	65.67	68.19	67.47	77.39	74.29	68.34	63.12
SVM (RBF)	91.43	91.26	87.49	86.76	97.97	97.91	92.26	91.81	90.54	89.57

Among the single classifiers, the linear discriminative models, Ridge regression and linear SVM, still have the top level performance. This is similar to what has been observed in previous experiments on the training set. Naive Bayes, SVM and kNN again perform on the same level. Given the fact that we do no carry out feature selection in this experiment, the observation about these three classifiers also agrees with pervious experiment. Compared with RBF kernel SVM's performance in the training set and test set experiments, the performance of kNN and the performance of Naive Bayes are respectively better and worse in test set than those in the training set. Simple decision tree classifier still generates significantly worse results than the other classifiers. See table 3.3 for details.

With regard to the ensemble methods, both stacking methods and the majority

voting get similar results while probability scores variant of stacking is worse than the other two methods for one test subset. Another observation is that none of the three ensemble methods provides higher F scores compared to the best single classifiers in each of the test sets. However, the generalization capability of ensemble methods is proved as they stably provide the top level performance.

Furthermore, no strong relationship between the performance of topic classification and the grade of privacy policies has been observed. This may partially be because of the fact that, in the datasets, only the paragraphs that can be manually sorted into the pre-defined categories are included. Hence, even if the privacy policies with lower grades may have less structured contents, the performance of topic classification will not deteriorate significantly. However, the numbers of categorized paragraphs for the privacy policies with lower grades may drop sharply, as shown in table 3.1.

In conclusion, the results of test datasets further support the observations and discussions in previous experiments on the training dataset.

3.4 Extensions

In this section, we introduce four extensions of the basic topic classification system so as to provide practical functionalities that provide adding value to the users.

3.4.1 Two-layer classification

We have introduced 16 categories that are used for topic classification in section 3.2. As observed in the datasets, the four core categories cover a significant larger portion of all paragraphs than other supporting categories do. The imbalance in the sizes of categories lead to the consideration about further classification of the paragraphs in some of the larger categories into more detailed sub-categories.

The definition of sub-categories depends on the common practice observed in privacy policies. For example, under the category of *'Sharing'*, there are paragraphs which cover different topics such as general statement of information sharing, information sharing with third party companies, information sharing during merge and acquisition and information sharing required by law. We provide the details of sub-categories as below:

Category of **'Collection'**:

- Category of **'User'**: User's information that is provided voluntarily by the user, e.g. during registration or form-filling.
- Category of **'Automatic'**: User's information that is collected automatically, e.g. user's online behavior and IP address, etc.
- Category of **'Others'**: User's information collected from other sources.

Category of '**Sharing**':

- Category of **'Statement'**: General statement about information sharing.
- Category of **'Third parties'**: Information sharing with third parties.
- Category of **'Law'**: Information disclosure required by law.
- Category of **'M&A'**: Information transfer under merger and acquisition.

Category of **'Choice and Access'**:

- Category of **'General'**: General information about user's choices and rights to access information.
- Category of **'Subscription'**: User's choices about subscriptions of emails, promotions, activities, etc.

Category of '**Cookies**':

- Category of **'General'**: General information about cookies.
- Category of **'Choice'**: User's choice of cookies, e.g. how to disable cookies and possible effects if cookies are disabled.

In general, we define 11 sub-categories under 4 categories. In order to integrate the sub-categories into the topic classification system, we propose a **two-layer classification structure**.

In the two-layer classification model, the basis classifier in the bottom layer is the classification model we discussed in section 3.3. The sub-category classifiers in the top layer are stacked on this basis classifier.

First, paragraphs in a privacy policy are categorized into the 16 top level categories by the basis classifier. Then, if a paragraph is classified into one of the four categories which have sub-categories, the corresponding top layer classifier is applied to label the paragraph with a sub-category.

All in all, with a two-layer structure, the system can provide the user with granular information. For example, a user, who is interested in information sharing

with third parties, can directly check this sub-category, saving the time spent on manually checking all paragraphs in 'Shareing' category.

The two-layer classification model has many other merits: 1) The overall performance is expected to be more stable than a flat structure where all sub-categories are considered as top level categories. 2) The two-layer classification structure provides strong flexibility as the top layer classifiers can be easily opted in or out. 3) Similarly, two-layer structure also contributes to more optimized computational efficiency due to the flexibility. 4) Furthermore, flexibility gives raise to user-friendliness, as the swifts of top layer classifiers can be implemented as user's preference options in the system.

Now we test the classification of the sub-categories within their own parent categories. To this end, we choose Ridge regression, linear SVM and Naive Bayes classifiers and the stacking method to test on the training dataset. The datapoints of the training dataset have been further labeled to reflect the sub-categories. Feature selection using χ^2 is applied to select 50% features in each case. We provide the test results generated from 50 runs of 10-fold cross validation:

Table 3.4: Sub-category classification - performance of top layer classifiers

	LSVM		Ridge		NB		Stacking	
	$F_{0.5}$	F_1	$F_{0.5}$	F_1	$F_{0.5}$	F_1	$F_{0.5}$	F_1
Collection	86.80	85.79	90.71	90.00	91.69	91.07	90.21	89.44
Sharing	93.14	92.84	94.19	93.95	94.81	94.58	95.05	94.68
Choice & Acc.	84.26	80.27	29.27	39.58	83.99	83.22	86.58	87.18
Cookies	93.72	92.25	39.34	50.66	97.39	96.02	94.74	95.07

The four sub-category classification tasks are all linear separable as shown in table 3.4. Naive Bayes performs surprisingly well in these tasks. One of the possible explanations is that the size of categories (2-4) is much smaller than the size of the main topic classification task (16). Ridge regression, however, encounters prbolem in precision scores for the last two sub-category tasks, where there are only two categories. We suspects that this is caused by implementation flaw in the applied library. Stacking method, the probability score variant, shows the merit of stable top-rank performance as concluded in previous section.

3.4.2 Multi-label classification

So far, in all previous discussions, we have assumed the single-label classification model, which always and only assigns the best label to a test item. This simplified model is suitable for paragraph-based classification of formal documents, such as well structured privacy policies. However, in real life, not all privacy policies are well structured. There are irrelevant paragraphs that cannot be sorted into any of the 16 categories and multi-purpose paragraphs that may be relevant to more than one category.

Therefore, multi-label classification is useful to improve the topic classification for non-ideal tset cases. In our system, we implement a multi-label mechanism that can both assign multiple labels and also assign no label if a test case is considered irrelevant. In current research stage, practical extensions are considered less essential than the core classification system. Thus, we only implement the multi-label functionality from engineering aspect without covering any advanced subject-specific techniques.

Our implementation of multi-label functionality depends on the actual computation process of the classifiers. For many classification models, such as Naive Bayes and SVM, confidence scores are computed for all categories in order to compare and decide the final label for a test item. These confidence scores can be either probabilities or signed distances to the hyperplane, depending on the specific scheme of the classifier.

Using this confidence score, we can control the behavior of the classifier so as to provide multi-label functionality. By running empirical tests, confidence score threshold t is set up for a specific classifier. For example, using manual labeled instances, a threshold t can be searched. For a test item, categories generate confidence scores higher than t are assigned as the class labels to the test item. Or if no category generates confidence score higher than t, no label will be assigned to the test item.

3.4.3 Privacy policy grading

Based on the results generated by topic classification system, a grading system can assist users to quickly comprehend the general quality of a privacy policy by providing an overall grade based on topic coverage.

3.4.3.1 Background

Agrawal et al. [4] study the issue of privacy policy ranking and propose a simple mathematical model for privacy policy grading. In their model, categories of user information type are predefined and sub-categories are allowed. The model itself is essentially a linear combination of weighted scores from categories. This type of linear model is expected to be effective and efficient. However, there are two problems with the proposed model. First of all, it depends heavily on natural language understanding, and the grading processes are carries out by manual inspection. This greatly reduces its practical value. Secondly, the model reflects the level of risks in a privacy policy. To be more precise, if a privacy policy expresses more undesired features, it will be graded with higher score and therefore considered as a worse privacy policy. However, such grading scheme will strongly surfer from privacy policies that simply do not provide enough information. It is also worth to point out that this paper is the only dedicated research on the issue of privacy policy grading to the best of our knowledge.

Closely related to our research in privacy policy grading, the study in automatic essay grading has also evolved largely due to the emerge of modern nature language processing and machine learning techniques. Back to the 1960's, research on automatic grading of essay [70, 111] began by studying the objectively measurable yet intuitive features from essays, such as length of essay in words, average sentence length, number of commas, and by finding the correlation between these features and the essays that receive higher human ratings. Later, the early research methods were replaced by more sophisticated methods during late 1990's and early 2000's, with several representative systems such as PEG (Project Essay Grade) [71], which applies multiple regression techniques, Bayesian essay testing system (BETSY) [87], which uses Bayesian networks, and the intelligent essay assessor [28], which is based on the Latent Semantic Indexing (LSI).

One of the challenges in building a grading system is that, in contrast to text classification, grading is a objective process by its nature, due to many subjective factors (such as personal preference, different background and understanding, or lack of clearly stated criteria or clearly quantified scaling criteria). For example, in a university course short answer grading experiment on the five-point scale, Mohler et al.[67] report exact agreement of scoring only by 56.8% percent with 17.0% scoring differs by more than one point and 3.0% differs by 4 points or more.

3.4.3.2 Grading scheme

Our proposed automatic privacy policy grading scheme is based on the topic coverage as measured by the core topic classification system. Particularly, two grading schemes are proposed to generate the grades for full-text privacy policies.

Our general privacy policy grading scheme is based on topic coverage, meaning that privacy policies are graded for their coverage of the predefined topics (categories). We believe this scheme is more effective and suitable for automatic privacy policy grading for the following reasons. Firstly, in contrast to the scheme proposed in [4], classification coverage conveys more accurate information about the general quality of privacy policy. That is to say, in general, we consider a privacy policy to have better quality if its contents are more thorough. Secondly, considering the main purpose of full-text privacy policy is to provide an overall estimation of the privacy policy so as to improve the awareness of the users, a generic metric is preferred over specific measures such as the measure of specific risks used in [4].

We apply a linear model to grade a privacy policy p based on its topic coverage:

$$Grade(p) = N(\sum_{i=1}^{n} w_i c_i)$$

where n is the number of all categories, w_i denotes the weight assigned to category i and $N(\cdot)$ is normalization function that normalizes the linear sum into a predefined scale, e.g. 10-point grading scale or 5-point grading scale. $c_i = 1$ when category i is covered in the privacy policy, else $c_i = 0$.

The grading model proposed above also provides a valuable functional flexibility – the weights w_i can be adapted from many sources, e.g. generated automatically by collaborative filtering, or predefined by users to better address his privacy preferences. Below we introduce two graders that rely on this grading model.

Simple grader We have presented a simple and effective privacy policy detector in section 2.2.1. The privacy policy detector uses predefined regular expressions to match contents in privacy policies. Derived from the similar regular expression scheme used in the privacy policy detector, the simple grader applies regular expression matching to decide topic coverage of a privacy policy. That is, if certain regular expression match happens, the grader considers the corresponding category is covered. Therefore, we derive a simple grader from the privacy policy detector.

Topic classification based grader Another type of grader is created based on the core topic classification system. To be more specific, the grader reads the classification results from the topic classification system, and then applies the linear model of grading to calculate the score for a privacy policy. Compared with the simple grader, the classification based grader is expected to be more accurate yet less computational efficient.

3.4.3.3 Evaluation

To compare the two types of graders, we carry out empirical evaluation using human judgment as a baseline. As already stated before, the challenge in evaluating grading is that it is more subjective than classification. We apply a baseline, which represents human reasoning in grading, in order to evaluate the graders. The graders are evaluated by measuring which grader provides closer grades to the baseline.

Setup To test the graders, we apply the 934 full-text privacy policy *dataset* as introduced in 2.1. This dataset is constructed using Google's search results on the keyword phrase 'privacy policy', which are manually classified as either privacy policy or not privacy policy. This dataset, consisting 796 privacy policies in total, forms the basis for the evaluation of graders in this section.

The two graders, denoted as $G1$ and $G2$, are applied on the 796 privacy policies. For each grader, we create three variants of the grading system by applying three different text extracting schemes. The three text extracting schemes are introduced in section 2.2.3. The variants are denoted as $G1_{v1}, G1_{v2}, G1_{v3}$ and $G2_{v1}, G2_{v2}, G2_{v3}$. This means, for each grader and privacy policy pair, three grades are generated in order to minimize the possible effects on the grades caused by preprocessing.

$$var_1 = var(G1_{v1}(p), G1_{v2}(p), G1_{v3}(p))$$
$$var_2 = var(G1_{v1}(p), G1_{v2}(p), G1_{v3}(p))$$
$$avg\text{-}diff = |avg(G1_{v1}(p), G1_{v2}(p), G1_{v3}(p)) - avg(G2_{v1}(p), G2_{v2}(p), G2_{v3}(p))|$$

For now, we have three sets of grades generated by each grader, i.e. grades generated by the three variants of each grader. First we calculated the statistical variance of the grades among the three variants for each grader, var_1 and var_2, and then we calculated the differences between averaged grades for two graders, avg.

In order to form a fair **baseline**, we only select the privacy policies that generate smallest var_1 and var_2 along with largest $avg\text{-}diff$ for the next step – manual grading. During the manual grading process, the human judge knows the linear grading model but does not know the grades generated by the two grader systems. Moreover, the human judge generates the grades based on the ruless of the linear grading model while also based on his general perception.

Result The human judge gives grades to 125 privacy policies based on the linear grading model along with direct instinct. Finally, we compare the average grades, which are averaged among three variants, from each grader against the baseline. To point out, all grades are normalized to a 10-point scale.

For 98 out of 125 (78.4%) test privacy policies, the grades generated by the topic classification based grader are closer to the human judge's grades than the grades generated by the simple grader. Also, measured against the baseline, simple grader's averaged difference is 1.68 ± 0.54 and topic classification based grader's averaged difference is 0.83 ± 0.34. Considering that we apply the 10-point scale, the grades generated by the latter are very close to the human judge's grades with less than one point diffidence on average.

3.4.4 Visualization

Besides the classification and grading extensions introduced so far, we also provide a visualization extension to visualize the results from topic classification.

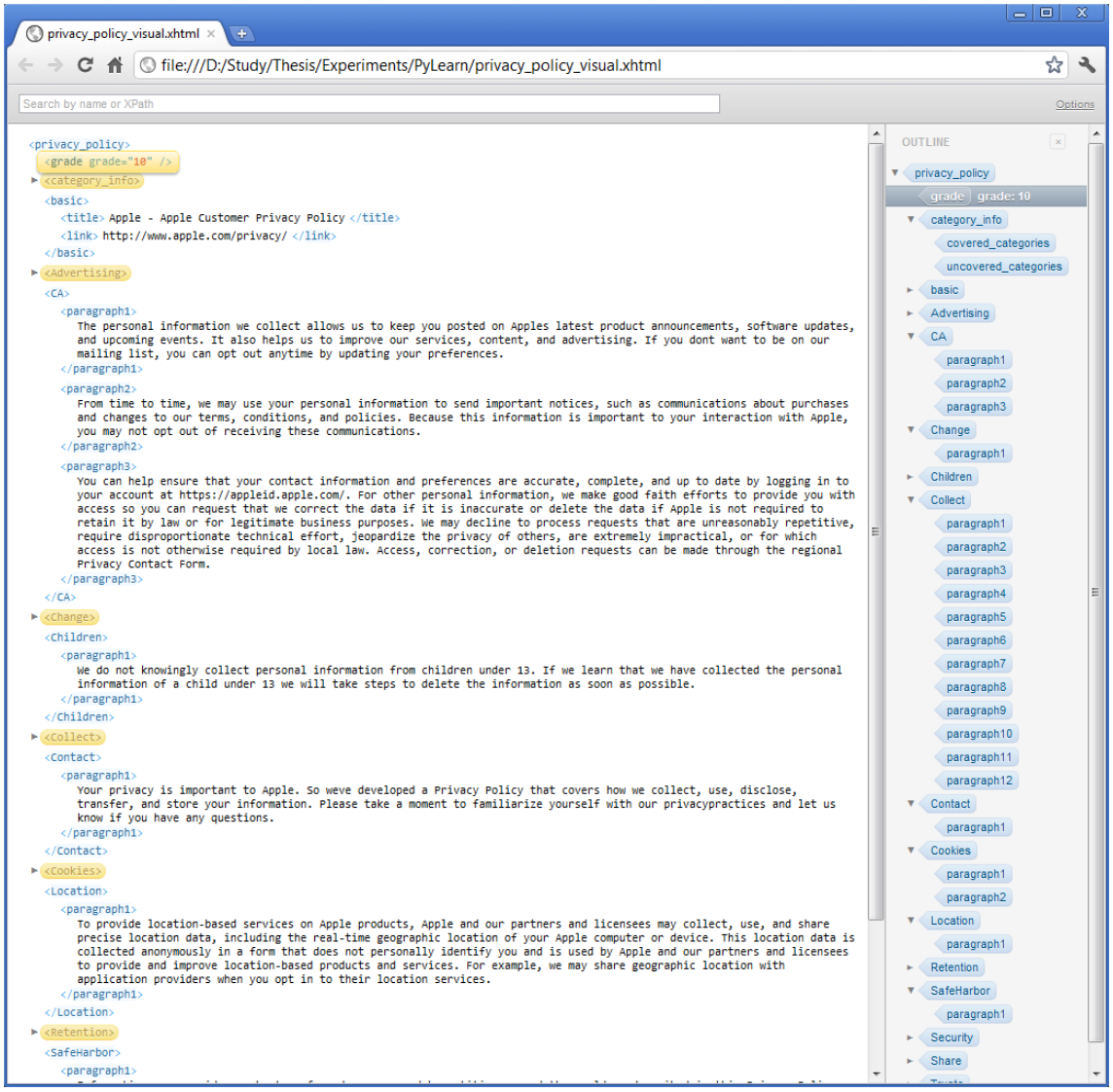
This extension depends on a Chrome add-on named XML viewer [13]. The visualization extension translates the privacy policy classification results into a .xml file. Then, the XML viewer is used to visualize this .xml file, see figure 3.6 for an example.

Though it is a simple user interface without elaborated design and implementation, this visualization method serves as a prototype to demonstrate some of the basic functionalities of visualization in our automatic privacy policy evaluation framework.

It demonstrates the topic viewing function, which assists the users in quickly browsing the privacy policies. Paragraphs of privacy policies are sorted into each specific topics. Hence the users can directly jump into any interested topic. This visualization function helps to shorten the time spent on finding the right informa-

[13] https://github.com/sergeche/xmlview

Figure 3.6: Visualization extension



tion that users concern and therefore improves the readability of privacy policies. In the tool, the navigation column on the right part of the screen facilitates the topic viewing function. Items in the content column are hidden by default. Upon click on either the entry in the navigation column or in the content column, the specific paragraph shows up.

Grading is also demonstrated in this visualization prototype. Grade of the privacy policy is placed right in the top of both the content column and navigation column.

3.5 Summary

In this chapter, we fulfill the research goal in building a topic classification system that applies the generic machine learning approach towards our privacy policy framework. The thesis covers the all aspects of the topic classification in a step-by-step manner, from the definition of categories at the very beginning to various extensions at the end. The classification results in the experimental evaluation support the idea of applying machine learning approach to automatically categorize, grade and analyze privacy policies.

Chapter 4

Share Statement Understanding

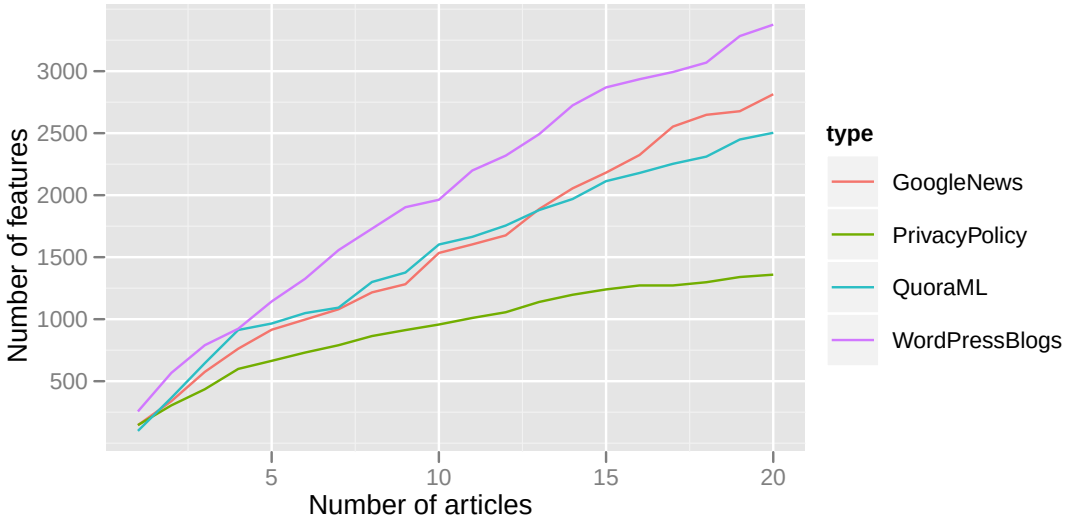
The basic topic classification approach has the deficiency of loosing the semantic information of the natural language statements. With the topic classification system presented in the previous chapter, it may happen that two similar privacy policies, where one states “We do not share your personal information in any condition” and the other states “We use, share and sell your personal information”, are graded in the same positive way.

This problem calls for fine-grained schemes that handle specific parts of privacy policies using the generic machine learning approach while taking the semantic information into account. The main research goal of this chapter is to go beyond the basic topic classification and explore the application of the generic machine learning approach to the task of understanding the real implication of specific statements in privacy policies.

Given the hypothesis that natural language understanding is AI-complete[1], there should be no ‘perfect’ solution for the task of automatic privacy policy understanding with the current state of the art. However, there are special patterns in privacy policies that are finitely classifiable. Therefore, it is possible to reduce the complexity of the task and create automatic systems for understanding the implication of privacy statements using our generic machine learning approach supported by text classification and semantic techniques.

[1]AI-complete is not a formal mathematical definition. It refers to the belief that for several of broad areas in AI, solving the problem of the area is equivalent to solving the entire AI problem, i.e. producing a generally intelligent computer program [96].

Figure 4.1: Privacy policy compared with other types of natural language articles



4.1 Overview

Specifically, privacy policies are special compared with common natural language texts and articles, such as Internet blogs, topic-specific discussions and news articles. As demonstrated in figure 4.1, privacy policies have limited amount of word features. Moreover, we observe strong expression formality and fixed patterns from the online privacy policies due to the legal nature of these policies.

Taking the advantage of the fact that privacy policies are well structured formal documents, we propose solutions to the task of automatic privacy policy understanding using two different approaches and carry out intensive empirical evaluation to demonstrate the performance of the systems based on these two approaches throughout this chapter.

As discussed above, the key of automatic privacy policy understanding is to reduce the complexity of the task by exploiting the special formality and patterns in privacy policies. There are many patterns in privacy policies, so that one could enumerate and tackle these patterns one by one. But in this chapter, we demonstrate a common methodology for exploiting the interesting patterns in privacy policies in order to understand the privacy policies using the generic machine learning approach. Hence, our task begins with choosing one representative pattern in privacy policies.

Share statement Out of many patterns in privacy policies, the *share statement* is particularly interesting. We select the *share statement* as the example pattern for two reasons. First, the share statements observed in hundreds of privacy policies are classifiable. Second, share statement is one of the most essential parts in privacy policies from user's perspective.

We define **share statement** as the high level statement in a privacy policy that declares the company's general attitude in sharing customer's information.

These *share statements* normally either come as the first paragraph in the 'How We Share Your Information' section or appear as the first sentence in one of the paragraphs in this section. Share statement is commonly a one-sentence text, but sometimes could be a two-sentences or three-sentences text.

In majority cases, a *share statements* answers the core question of "what is the scope of sharing?". More specifically, this scope is about "Does a company share user's information", "If user's information may be shared, is the company going to ask for user's consent beforehand?", "What are the exceptional situations in which a company will disclose user's information?", etc.

Three examples of share statements extracted from three different privacy policies, each describing different intentions related to the disclosure of personal information, are listed below:

> "We may sell, rent, license, trade or otherwise disclose your personal information, including but not limited to your mailing address, phone number and other registration information." – Hanleywood's Privacy Policy

> "As a general rule, Blizzard will not forward your information to a third party without your permission." – Blizzard's Privacy Policy

> "Except as described in this statement, we will not disclose your personal information outside of Microsoft and its controlled subsidiaries and affiliates without your consent." – Microsoft's Privacy Policy

These three examples also reflect the deficiency of the basic topic classification in previous chapter. Assuming these privacy policies get a same grade, without a fine-grained scheme in understanding the share statements, we receive no further evaluation from the system about the huge difference in the share statements.

Table 4.1: Categories of share statements

3-class	8-class	Demo sentence
Positive		
	Not sell Not share	We do not sell, rent or share your personal information to any third parties.
	Not sell	We are not in the business of selling or leasing your personal information.
Neutral		
	Share only under consent	We do not share your information without your consent.
	Share only under consent with exceptions	We only share your personal information with your permission or under following circumstances as described in this privacy policy.
	Share for exceptions	We respect your privacy and only share your information for the following limited reasons.
	Not share for marketing purpose	We will not share any of your information with third parties for marketing purposes.
Negative		
	No limit share	We will share your information with our partners.
	Sell and share	We reserve the right to sell, rent and otherwise disclose your information to but not limited to the following third parties.

Task formulation It is clear to a human judge that these three *share statements* above have different scopes concerning the disclosure of user's information. That said, it is not trivial for a computer system to automatically understand the differences among these share statements. Even more, a formal interpretation of the task of 'share statement understanding' is necessary before any meaningful discussion.

To define the task of 'share statement understanding' in a way that computer can tackle, we limit the scope of the problem and reduce it to a text classification task. In other words, our methodology for 'share statement understanding' is to first predefine categories from observed patterns of share statements, and then build the systems that classify share statements into these categories.

Categories of share statements We summarize the categories of share statement in table 4.1. These categories are defined from the observation on two hundred privacy policies. Based on these categories, the task of 'share statement understanding'

is then formalized into two sub-tasks.

The first sub-task is to classify any given share statement into one of the eight classes. The eight categories provide fine-grained information about the disclosure practice of the company. These categories are listed from the most positive to most negative in table 4.1.

Similarly, the second sub-task is to classify share statements into three high-level categories, that are, positive, neutral and negative. Table 4.1 provides the mapping between eight categories and three categories as well.

Throughout this chapter, we will use the terms '3-class task' and '8-class task' to refer to these two sub-tasks.

It is important to point out a few special features of these categories.

Firstly, the eight categories are not externally exclusive. In other words, there are share statements that do not fall into any one of the eight predefined categories. However, as shown in table 4.3, in a survey of 95 privacy policies, we identify only 6 (6.32%) of such share statements. This is a proof of the good conclusiveness of our categories.

Secondly, the eight categories are not strictly mutually exclusive. Though it is very unlikely that it belongs to another category if a share statement falls into one of the categories, exceptions still exist. The overlaps may happen between the 'Not sell information' category and other categories that only describe the practice about information disclosure.

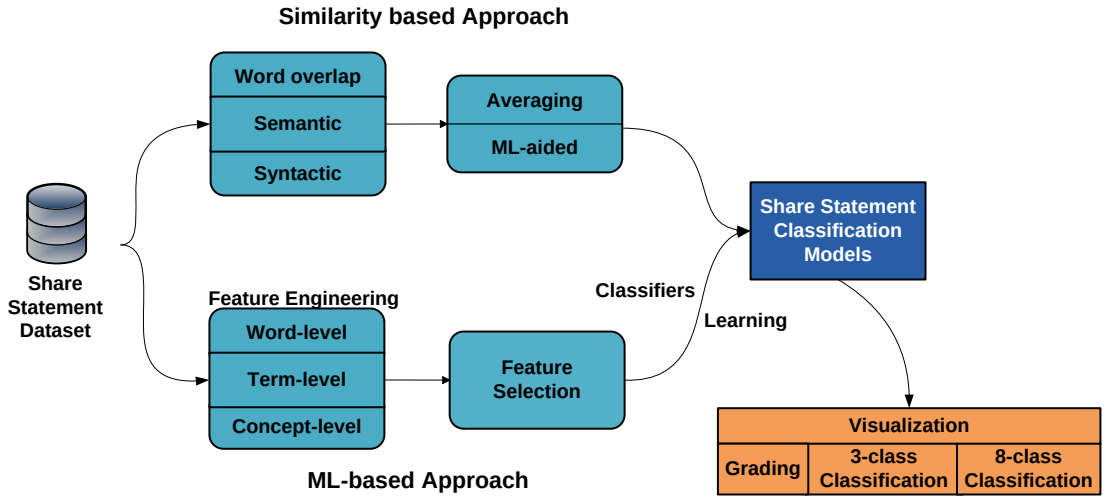
Thirdly, though the eight categories are supposed to be ranked from most positive to most negative from the top to the bottom in the table 4.1, the interpretation of this ranking is subject to personal interpretation.

Chapter outline In this chapter, we cover two approaches for solving the 'share statement understanding' problem. A general overview of these two approaches is drawn in figure 4.2.

Similarity-based approach This approach uses similarity between sentences to decide whether a given pair of sentences have the same meaning. By matching test sentence with labeled sentences and measuring the similarity, label is given to the test sentence based on the closest matches.

Machine learning-based approach This approach directly applies text classifiers on the sentence dataset and learns classification models that will automatically

Figure 4.2: Share statement understanding overview



separate 'share statements' into different meanings.

4.2 Similarity based approach

All the techniques in this section calculate a similarity score $Sim(Q, R)$ between two sentences Q and R. By calculating this similarity score, such techniques intend to capture numerically the extent to which two sentences convey the same information. The objective is to calculate $Sim(Q, R)$ for all sentences R in a collection and know that when the score Sim is maximized, the sentence R has a high degree of similarity to the query sentence Q.

4.2.1 Word overlap measures

Metzler et al. [64] apply a baseline word overlap measure in their empirical research. We follow their choice to use the simple word overlap measure as a baseline for the comparison of more complex measures and the comparison with the machine learning based approach.

Simple word overlap measure Simple word overlap fraction ($Sim_{overlap}$) is defined as the proportion of words that appear in both sentences, normalized

by the sentence length. $Sim_{overlap}$ is denoted as:

$$Sim_{overlap}(Q, R) = \frac{|Q \cap R|}{|Q|} \tag{4.2.1}$$

where $|Q \cap R|$ denotes the number of terms that appear in both sentence Q and R.

IDF overlap measure This simple baseline measure can be extended to an IDF variant by using weights of inverse document frequency on the proportion of words that appear in both sentences.

By taking IDF into account, the variant measure reflects the fact that terms with different IDF contribute differently to the similarity score between two sentences. Generally, a term with higher IDF, meaning the term appears in less documents in the whole collection, is considered as a stronger indication of similarity. The $Sim_{overlap\_idf}$ is defined as:

$$Sim_{overlap\_idf}(Q, R) = \frac{|Q \cap R|}{|Q|} \left(\sum_{w \in Q \cap R} \log \frac{N}{df_w} \right) \tag{4.2.2}$$

where N denotes the number of documents in the collection and df_w is the number of documents that w appear in.

Phrasal overlap measure The word overlap measures introduced above treat a sentence simply as a bag of words and only capture the overlap between words. This is a significant limitation because possible phrasal overlaps can also contribute to the similarity between sentences. Therefore, further improvement could be achieved by differentiating between the word overlaps and the phrasal overlaps and taking phrasal overlaps into account.

Banerjee et al. [5] propose a gloss overlap measure, called *extended gloss overlap*, that takes phrasal overlap into account. The base argument of this measure is that the relationship between the lengths of phrases and their frequencies in the text collection follows the Zipfian distribution. Hence, the overlap score between two sentences, $overlap_{phrasal}(Q, R)$, can be computed as $\sum_n m^2$ for n phrasal m-word overlaps to approximate the Zipfian distribution.

To normalize this score into the range of $[0, 1]$, Ponzetto et al. [78] propose the following scheme, which first normalizes the overlap score with sum of two

sentence lengths and then takes the value of the hyperbolic tangent function in order to minimize the effect of outliers:

$$\begin{aligned} Sim_{overlap\_phrasal}(Q, R) &= \tanh\left(\frac{overlap_{phrasal}(Q, R)}{|Q| + |R|}\right) \\ &= \tanh\left(\frac{\sum_n m^2}{|Q| + |R|}\right) \end{aligned} \tag{4.2.3}$$

4.2.2 Semantic measures

WordNet WordNet [26] is a lexical database which groups English words, more specifically, nouns, verbs, adjectives and adverbs, into sets of cognitive synonyms called synsets, each expressing a distinct concept, and provides short, general definitions for these concepts. Each sense of a word is in a specific synset, which can be viewed as a structure containing sets of terms with synonymous meanings.

WordNet defines the relations between synsets and the relations between word senses, denoted as semantic relations and lexical relations respectively [22]. The semantic relations connect synsets from one to another. Hypernym-hyponym ('is-a') relations for nouns[2], hypernym-troponym[3] for verbs, holonymy ('has-a') and meronymy ('is-a-part-of') are all semantic relations defined in WordNet.

Given these relations among synsets defined, WordNet actually provides a huge net of synsets (senses of words), in which paths exist between synsets (word senses). These paths between synsets can be measured to reflect the semantic similarity.

4.2.2.1 Word-level semantic similarity measures

There are many proposals for measuring the semantic similarity between two synsets in the context of WordNet. Below, we briefly cover several commonly used semantic similarity measures. In contrast to our goal, which is measuring the sentence-level similarity, the similarity measures in this subsection are about word-level similarity. In order to avoid confusion, we denote the word-level similarity measures in this subsection using $Synset\_Sim(s, t)$, where s and t are two senses of words (two synsets in WordNet).

[2]For example, in sentence 'Red is a color.', red is a hyponym and color is a hypernym.

[3]Troponymy is a coined term by WordNet authors as an analogy to hyponymy for nouns. The reason troponymy is not exact hyponymy is that for nouns the 'is-a' relationship is obvious but verbs are not subject to straightforward 'is-a' relationship. For example, it is clear to say 'red is a color', but it is not straightforward to say 'whispering is talking'.

To begin with, we define *length* as the length of the shortest path between two synsets, *D* as the maximum depth of a taxonomy[4], ***least common subsumer (LCS)*** as the shared parent of two synsets that does not have any child that is also a parent of the two synsets, *P(s)* as the probability of encountering an instance of synset s in a large corpus, and ***Information Content (IC)*** as $IC(s) = -\log P(s)$.

Path Distance Similarity is the simplest way to calculate the word-level similarity using WordNet:

$$Synset\_Sim(s,t)_{path} = \frac{1}{length}$$

Leacock Chodorow Similarity [53] extends the length-based measure and scales the length by the overall depth D of the taxonomy:

$$Synset\_Sim(s,t)_{LCH} = -\log \frac{length}{2D}$$

Wu-Palmer Similarity [112] defines the similarity using the addition between the depths of synsets and the depth of their least common subsumer (LCS):

$$Synset\_Sim(s,t)_{WUP} = \frac{2depth(LCS)}{depth(s) + depth(t)}$$

Resnik Similarity [85] explores the usage of information content in semantic similarity measure:

$$Synset\_Sim(s,t)_{RES} = IC(LCS)$$

Jiang-Conrath Similarity [46] further extends the information content method as:

$$Synset\_Sim(s,t)_{JCN} = \frac{1}{IC(s) + IC(t) - 2IC(LCS)}$$

Lin Similarity [55] is another form of the information content based measure as:

$$Synset\_Sim(s,t)_{LIN} = \frac{2IC(LCS)}{IC(s) + IC(t)}$$

Mohler et al. [67] and Budanitsky et al. [13] provide comparisons of these similarity measures based on detailed experiments. Generally, information content based Jiang-Conrath similarity performs the best overall, followed by the similar information content based Lin similarity [13]. Under the context of applying similarity for short question answering, Jiang-Conrath similarity generates the best results too, but surprisingly, followed by the simplest path distance similarity which generates almost the same performance [67].

[4]For WordNet taxonomy the maximum depth is 16 if we presume all the hierarchies have a common parent node.[113]

4.2.2.2 Applying semantic similarity measure to the sentence-level

Despite many word-level semantic similarity measure introduced above, it is not trivial to create a scheme for calculating the semantic similarity between two sentences using the WordNet word-level similarity measure.

To produce the sentence similarity score, we need to figure out which word sense from WordNet to apply for a given word in its sentence context. This leads to the fact that the sentence-level similarity calculation based on WordNet word-level semantic similarity requires the solution of the word sense disambiguation (WSD) problem. However, WSD is still an open research topic in natural language processing and it is an AI-complete problem [68].

From implementation perspective, it is not validate to directly apply sentence level similarity measure without considering the issue of WSD. Because each word may have more than one senses, most of which are not necessarily relevant to the meanings intended in the sentences. Therefore, the performance of a similarity measure suffer when these unrelated senses cascade and produce meaningless similarity scores.

Hence, before building a sentence-level similarity calculation method on top of the word-level similarity measures, we need to find a workaround of the WSD problem.

We apply two common and straightforward workarounds of the WSD problem:

- First sense heuristic

  Also called predominant sense heuristic. The first sense heuristic method simply chooses the first sense from WordNet for any ambiguous word for computing the similarity between sentences. Though simple in nature, it frequently outperforms some dedicated WSD methods even when they take surrounding context into account [61]. The downside of the first sense heuristic is also obvious. It does not take sentence context into account. It relies on the quality of first senses from WordNet and presumes the first senses are reasonable in most of the cases.

- Internal similarity maximization

  The internal similarity maximization method maximizes the internal semantic similarity before calculating the similarity between two sentences. That is, for each ambiguous word, it chooses the sense that can maximize the overall

similarity of the senses in the sentence. For implementation, brute-force method is applied to enumerate all possible combinations of senses in the sentence and then select the combination which has the highest similarity score.

4.2.2.3 Sentence semantic similarity

Related work There are many proposed methods that calculate sentence-level similarity measure by applying WordNet word-level similarity measure, such as cosine similarity measure between semantic vectors of two sentences [54], inter-sentence maximum similarity measure[5] with IDF weights [65], and simplified inter-sentence maximum similarity measure [56].

Achananuparp et al. [2] carry out extensive experiments of various sentence similarity measures using three benchmarks[6]. Based on this empirical research, the simplified maximum similarity measure proposed by Malik et al. [56] constantly performs on the top for the task of sentence similarity measuring. Hence, we apply this method to represent the various WordNet-based semantic similarity measures with simplest first sense heuristic method as a baseline.

Below, we give a brief description of the two selected methods, the first sense heuristic method and the Malik et al. method.

First sense heuristic measure First sense heuristic directly selects the first sense for each ambiguous word in both sentence Q and sentence R. Further, the method iteratively computes the word-level similarity, using any one of the word similarity measures introduced in 4.2.2.1, between two terms s and t each drawn from Q and R. It means, in this step, similarity of all possible combinations of two word pair from sentences Q and R are calculated.

To get the similarity score for two sentences, the final step is to form a scheme in aggregating and normalizing the similarity scores of all possible combinations of two word pair. To this end, we propose a parameter α in the calculation of the similarity score. In this final step, $\alpha * 100\%$ combinations with highest scores are selected out of all combinations. And scores of these selected combinations are

[5] Inter-sentence maximum similarity measure is a coined term in this report to denote the scheme as introduced later.

[6] TREC9 (The Ninth Text REtrieval Conference), MSRP (Microsoft Research Paraphrase Corpus) and RTE3 (Third Recognising Textual Entailment Challenge).

summed up and then normalized by the number of selected combinations to form the final sentence similarity score:

$$Sim_{first\_sense\_\alpha}(Q, R) = \frac{\sum (top\ \alpha\ highest\ Synset\_Sim(s, t))}{\alpha\ |\{Q\} \times \{R\}|} \tag{4.2.4}$$

where $\{Q\}$ denotes a set of all words of sentence Q, $\{Q\} \times \{R\}$ denotes all possible combinations of word pairs with the first element s selected from Q and the second element t selected from Q, and α is the parameter within the range [0, 1] that defines how many percent of highest similarity scores are used.

The reason for using parameter α to select word pairs with highest scores is that not all word pairs have semantic meaning to be combined together. Therefore such top $\alpha * 100\%$ selection process automatically eliminates the side-effect from the meaningless word pairs. Due to this reason, the top $\alpha * 100\%$ selection is also expected to outperform random $\alpha * 100\%$ sampling.

Though we expect first sense heuristic measure with top $\alpha * 100\%$ selection will outperform a measure using scores from all word pairs or random $\alpha * 100\%$ sampling, there is another prospective in selecting the useful word pairs, that is to use only the types of words that normally carry the semantic contents.

Hence, we propose a variant of first sense heuristic measure that selects the terms from sentences using POS tagging. Specifically, before any word-level similarity calculation is done, the two sentence term-sets $\{Q\}$ and $\{R\}$ are first reduced to $\{Q_{POS}\}$ and $\{R_{POS}\}$ with only terms that have the POS tags in interest. In practice, we only preserve four types of POS tags, namely, nouns, verbs, adjectives and adverbs. For this variant, the similarity scores are computed as:

$$Sim_{first\_sense\_POS}(Q, R) = \frac{\sum_{\{s,t|tag(s),tag(t)\in\{POSset\}\}} Synset\_Sim(s, t)}{|\{Q_{POS}\} \times \{R_{POS}\}|} \tag{4.2.5}$$

where $\{POSset\} = \{nouns, verbs, adjectives, adverbs\}$[7] and $pos(s)$ the POS tag of term s.

Inter-sentence maximization measure The inter-sentence maximization measure shares much similarity with the internal similarity maximization scheme for WSD. The major difference is that inter-sentence maximization measure maximizes the word-level similarity for each word in a sentence by iterating and computing

[7] In implementation, POS tagsets are specified in more details, the UPenn Treebank tag set is often used. See http://www.cnts.ua.ac.be/pages/mbsp-tags and [59].

the similarity scores with all words the other sentence. Put differently, the core of inter-sentence maximization measure for two sentences is a WSD scheme. This WSD scheme disambiguates word senses by choosing the word sense that maximizes the word-level similarity between the word itself and any other words in the other sentence.

Given two sentences Q and R, inter-sentence maximization measure maximizes the word-level similarity score for each word s in Q with the words in R. Due to the fact that in WordNet synset similarity measures do not cover cross-POS, the iteration process above is limited to calculate only between two words in the same POS class in order to reduce the computing time.

The original method, first proposed by Mihalcea et al. [65], also takes IDF into account for normalizing maximized similarity score for each word. However, empirical study [54] shows that a simplified version of the method by Malik et al. [56] performs even better without taking IDF into account. We will apply Malik et al. for further experiment:

$$Sim_{inter\_sentence\_max}(Q,R) = \frac{\sum_{s\in Q} max\ Synset\_Sim(s,r)}{|Q|+|R|} + \frac{\sum_{t\in R} max\ Synset\_Sim(t,q)}{|Q|+|R|} \tag{4.2.6}$$

where lower-case q and r denotes all terms iterated over sentences Q and R.

4.2.3 Syntactic measures

All semantic similarity measures introduced to this point rely on the word-to-word lexical similarity based on WordNet. In other words, these methods use the 'bag-of-words' representation.

Compared with the word overlap measures in section 4.2.1, the lexical semantics is expected to be superior. However the lexical semantic measures do not contain all information conveyed by the sentence as, syntactic information such as the the order of words is not preserved by the 'bag-of-words' model.

Word order measure To take the syntactic information, specifically the word order, into consideration, Li et al. [54] propose a scheme to calculate the word order similarity between sentences based on vector representation. We describe this method below.

For sentences Q and R, a joint set J is defined as a bag-of-words of all terms that occur in both Q and R. Once this joint set J is formed, each term in J is binded with a number that reflects its order in J. For example, Q = 'Red is a color.' and R = 'Blue is a color.', then J = {(red,1), (is,2), (a,3), (color,4), (blue,5)}.

After a joint set J is formed based on sentences R and Q, two word order vectors $\mathbf{r}_Q$ and $\mathbf{r}_P$ are computed. The lengths of $\mathbf{r}_Q$ and $\mathbf{r}_P$ are the same as the number of terms in J. The $\mathbf{r}$, e.g. $\mathbf{r}_Q$ is calculated by iterating over all terms in J in the following manner:

For the ith word w_i in J, check if it exists in Q. If this w_i exists in Q and it is at the kth position in Q, then set $\mathbf{r}_{Q_i} = k$. If this word does not exist in Q, iterate over all words in Q to find if there exists a word that has a similarity with w_i higher than a given threshold β. If so and assume this word in Q is located as the mth term in Q, still set $\mathbf{r}_{Q_i} = m$. However, if during the iteration, no word in Q has similarity with w_i higher than the threshold β, then we set $\mathbf{r}_{Q_i} = 0$.

To continue with the example above where J = {(red,1), (is,2), (a,3), (color,4), (blue,5)}, $\mathbf{r}_Q$ is {1 2 3 4 0} and $\mathbf{r}_R$ is {0 2 3 4 1} if similarity between 'red' and 'blue' is lower than the threshold β, or $\mathbf{r}_Q$ is {1 2 3 4 1} and $\mathbf{r}_R$ is {1 2 3 4 1} if higher.

Given two sentences Q and R along with $\mathbf{r}_Q$ and $\mathbf{r}_R$ calculated following the above scheme, the word order similarity between the two sentences are defined as:

$$Sim_{word\_order}(Q, R) = 1 - \frac{\|\mathbf{r}_Q - \mathbf{r}_R\|}{\|\mathbf{r}_Q + \mathbf{r}_R\|} \tag{4.2.7}$$

Hybrid measure A hybrid measure that combines the measures of both lexical semantic and syntactic similarity will benefit from both sides. Li et al. present a linear combination. Empirical research [3] shows that the linear combination sentence similarity measure performs well when the semantic measure part is weighted more significant than the syntactic measure part.

We define the hybrid measure as below:

$$Sim_{hybrid}(Q, R) = \delta Sim_{semantic}(Q, R) + (1 - \delta) Sim_{syntactic}(Q, R) \tag{4.2.8}$$

where $Sim_{semantic}(Q, R)$ is one of the three semantic measures introduced in section 4.2.2.3 and $Sim_{syntactic}(Q, R)$ is the word order similarity described above.

4.2.4 Applying sentence similarity measure for classification

Given the sentence level similarity measures described in previous sections, the next step towards a final working similarity based scheme is the utilization of the similarity scores computed between sentences in the classification tasks. Though not the core of the similarity based approach, this step is not trivial and can highly effect the final models.

In this section, we propose two methods that apply sentence level similarity measures to solve the share statement classification tasks. These two methods differ in their ways of computing the final prediction based on a given 2d matrix calculated from the training set and the test items. Before we go into the details of these two methods, we will first start with a brief description of related work.

Related work Zhou et al. [120] present a classification algorithm based on semantic similarity. However, their method is based on the generalized vectors of classes, which are not available in our case. Mohler et al. [67] use semantic similarity measure to classify short answers into grading scales. However, this paper does not cover any explanation of the classification method that applies the similarity measures between short texts.

Two solutions The two methods that apply similarity measures for the classification tasks are formed on a common basis. To start with, we assume there are one training set with size n and one test set with size m. A $m \times n$ matrix of similarity scores is computed based on these two sets. More specifically, each row in the matrix corresponds to the similarity scores for one particular item in the test set and consists of the similarity scores computed between this one test item and all items in the training set. This means that the $m \times n$ matrix hosts the similarity scores of all pair-wise combinations of the training items and the test items.

Assuming we choose one of the similarity measures described in previous subsections 4.2.1, 4.2.2 and 4.2.3, we can calculate a $m \times n$ matrix of similarity scores using the method introduced above. With this matrix, we can further generate the classification predictions using the two methods proposed below, namely, the **averaging method** and the **machine learning aided method**:

Firstly, a straightforward yet effective method is to apply an averaging scheme on the matrix. For each test item, an average similarity score is computed for each category in the training set using the arithmetic mean. Assuming there are k

categories in the training set, k averaged similarity scores will be calculated. For prediction, the test item is assigned to the category which has the highest averaged similarity score.

Despite its straightforwardness and effectiveness in general cases, we do expect this method to have some difficulties in our specific share statement tasks. The main reason is that, in our specific tasks, sentences from different categories may share very similar vocabulary set. This leads to very close averaged similarity scores among some categories and gives raise to inaccuracy in prediction.

To amend this disadvantage, we propose a second method that applies machine learning technique for predicting the categories of test items. Besides the $m \times n$ matrix, this method requires another $n \times n$ matrix to be pre-computed as well. This $n \times n$ matrix is calculated based on the training set itself, meaning the $n \times n$ is a similarity matrix of the training set.

Using both the $n \times n$ matrix and $m \times n$ matrix as input, the machine learning method learns on the $n \times n$ matrix, which is computed solely from the training set, and then predicts on the $m \times n$ matrix. To be more specific, the learning process uses the rows in the $n \times n$ matrix along with the labels for each row as inputs. And the predict process uses the learned model to predict the labels for the rows in $m \times n$ matrix.

The machine learning aided method is expected to discover the common patterns of the similarity arrays and therefore provide better generalization and more accurate predicting results than the averaging method. It is also clear that the computational cost of the machine learning aided method is $\frac{n}{m}$ times higher than the averaging method.

4.3 Machine learning based approach

In this section, we explore the direct application of text classification techniques to the task of share statement understanding. Classification algorithms with the basic setup are first tested. Further, various feature engineering techniques are studied. Empirical evaluations are carried out in later section 4.4.

Related work in short text classification Though machine learning techniques have dominated the field of modern text classification for more than a decade, there was limited focus on short text classification task. Compared with general text

classification tasks, the short text classification task has its own distinct features. Back to 2000, Zelikovitz et al. [116, 115] study the short text classification using unlabelled background knowledge, and later they apply the latent semantic index [117]. Healy et al. [40] study the short text classification using case base reasoning. It is just after the raise of social networks, especially the microblogging services, short text classification has gain more attentions, for the applications such as text classification and sentiment analysis on Tweets [36].

4.3.1 Classifiers

To begin with, we apply Ridge regression, Naive Bayes, k Nearest Neighbours, linear SVM with l2 regularization and decision tree.

For Ridge, given its superior performance in previous chapter for topic classification, we expect it to have higher than average performance for the text classification task in this chapter.

SVM is widely used for text classification tasks and in practical systems with constant top tier performance. We use linear SVM with l2 regularization.

Naive Bayes has one of its strong advantages in its efficiency. Shown in previous chapter, Naive Bayes can perform in the top rank with feature engineering techniques, which is an very interesting feature for our discussion in this chapter. As stated in previous chapter, it has been shown that the Multinomial model is usually superior to the Bernoulli model in general text classification tasks [60]. However, in case of classifying short texts, Bernoulli model has better chance in matching the Multinomial model due to the fact that, in short documents task, whether a word occurs may be more important than how many times it occurs. We therefore apply the Bernoulli model in this section.

kNN classifier, however, is expected to have poor performance in this task. One of the reasons for this expectation is that imbalance in the categories will lead to more misclassification for kNN even when k is a small integer.

Given the pattern of share statement, decision tree could achieve reasonable performance. Though we do not expect to achieve top level performance with a single decision tree without ensemble methods.

4.3.2 Feature engineering methods

In normal text classification tasks, it is clear that using words, rather than more complex engineered features, as a vehicle to capture the information from texts in general text classification tasks is well justified. However, applying NLP engineered methods is beneficial in our specific task of share statement understanding. The reason is twofold. Firstly, this specific task requires finding statistical pattern from short texts, very limited amount of features and train samples. Secondly, our task cannot be well solved by 'bag-of-words' representation using normal feature setup, due to the substantial loss of linguistic information.

To tackle our task of share statement understanding, elaboration on feature engineering is even more important than selecting the classification algorithms, due to the nature of short text classification. In this section, we intensively study feature engineering techniques in the context of our specific task. These techniques can be further grouped into three types:

Word-level feature engineering Word is the basic unit to form sentences and documents. Words can be directly extracted from documents to form the traditional bag-of-words representation, which is an unordered collection of words. We study the word-level feature engineering methods, such as stopword elimination, stemming and lemmatization.

Term-level feature engineering Terms are single words and multi-word phrases selected from the corpus. For term level model, we experiment n-gram, term extraction with POS filtering and full-packaged term extraction.

Concept-level feature engineering Concepts are features that preserve the semantic and/or syntactic information, which is left out by t he bag-of-words model. For example, by using POS tags, the word 'book' can represent different concepts such as the noun form, meaning 'a collection of written sheets of paper' or the verb form, meaning 'to reserve for future use'. Besides the variants of POS tagging, we further investigate different ways of applying WordNet knowledge base to form the bag-of-synsets representations in order to preserve more semantic information.

In table 4.2, we summarize the differences among word-level, term-level and concept-level representations and demonstrate the results from various feature engineering techniques.

Table 4.2: Comparison of different feature engineering techniques

Methods	Results
Original sentence	These lazy geese and dogs can not jump over the quick brown fox.
Stopword removal	Lazy geese dogs jump quick brown fox.
Stemming	These lazi gees and dog can not jump over the quick brown fox.
Lemmatization	These lazy goose and dog can not jump over the quick brown fox.
Netgation bigrams	These lazy geese and dogs can not jump over the quick brown fox. 'not jump'
Term extraction (POS)	Lazy geese dogs not jump quick brown fox.
Term extraction (Full)	fox geese dog
POS tags	DT JJ NN CC NNS MD RB VB IN DT NN NN NN
POS tagged words	These/DT lazy/JJ geese/NN and/CC dogs/NNS can/MD not/RB jump/VB over/IN the/DT quick/NN brown/NN fox/NN.
Synsets (first sense)	These lazy,s,01 goose,n,01 and dog,n,01 can,n,01 ot,r,01 jump,n,01 over,n,01 the quick,n,01 brown,n,01 fox,n,01.
Synsets (int-max)[1]	These lazy,s,01 fathead,n,01 and cad,n,01 buttocks,n,01 ot,r,01 jump,n,06 over,n,01 the quick,n,01 brown,n,02 fox,n,05.

1: The variant 2 is applied for the internal maximization method of WSD.

4.3.2.1 Word-level feature engineering

Stopword removal Common word removal is widely used for text classification tasks in English. These words are overly common that they occur too often to convey essential information that is discriminating to separate apart documents in categories. There are two ways to eliminate these words, using a threshold of occurance or setting up a customized stopword list. The threshold approach is unbiased but not domain specific. The stopword list approach is language specific but provides less flexibility.

We include four stopword removal strategies in our experiment: 1) common English stopword list[8] approach., 2) threshold approach, 3) self-defined stopword list and 4) no stopword removal.

Stemming and lemmatization Stemming and lemmatization are all text normalization techniques. Stemming is the process of merging various word forms, such

[8]As defined in Appendix C.3

as plurals of nouns and conjugations of verbs, into their stems. Lemmatization only finds the strict lemmas of words. Stem is not necessarily the same as the morphological root of the word. This differentiates these two nearly interchangeable processes, stemming and lemmatization. For instance, for the conjugations of verb 'go', i.e. 'goes', 'going', 'went' and 'gone', stemming produces 'goe', 'go', 'went' and 'gone', while lemmatization yields 'go' correctly for all cases.

However, for English, which is relatively a less inflected language, stems are normally close to or same as the lemmas. This fact guarantees general quality of stemming for the purpose of text normalization and makes stemming widely accepted in diverse fields of applications. To point out, this slight difference of strictness also makes lemmatization a harder task than stemming [103].

In the experiment section, we test both stemming and lemmatization for text normalization.

4.3.2.2 Term-level feature engineering

As mentioned previously, we also explore the term-level feature engineering by either extending the bag-of-words model with phrases and multi-word terms or extracting terms from the bag-of-words model.

Negation bigram term construction Negation plays a crucial role in sentiment analysis [109, 108]. Our task of share statement understanding has a direct resemblance to sentiment analysis, in the way share statements can be categorized as positive, neutral and negative. For example, the share statement "We neither rent nor sell your personal information to anyone." is consider much more positive than "We may disclose your personal data to third parties."

To exam the role of negation in our task, we present a scheme for constructing bigram negation terms. The basis of this negation bigram construction scheme is that there are several common patterns of negation as observed share statements. For instance, when expressing sharing with limits or no sharing, patterns such as "... not share, sell or rent ..." and "... never rent, sell or share" are used with very high frequency. We describe the negation bigram construction scheme below:

$$P = (\ \{negation\,words\},\ w,\ \{POS\,tags\}) \tag{4.3.1}$$

P is a negation bigram construction pattern which is defined as a tuple of the

negation word set, a window $w = (w_{pre}, w_{pos})$ and the POS tag set. Such a pattern P defines a set of negation words along with its influencing distance, which is defined by the window w, and its influencing POS tags. For example, a pattern $P = (\ \{not, never\},\ (0, 10),\ \{VB\})$ matches any verb that appears within 10-word distance after a negation word 'not' or 'never'. For a match, such as a verb 'sell' that appears two words after 'not', a bigram negation term will be constructed, in this case 'not sell'.

The negation term construction begins with the definition of the negation word set and patterns. Then, the input sentences are easily processed to construct the negation bigram terms. For a certain input sentence, it is screened by looking up a dictionary of the words in the negation word set. For any match of negation word, POS tags are checked within the window in order to find all possible matches. Finally, negation bigram terms are formed for all matches.

n-gram n-gram is a contiguous sequence of n items from a given sequence of text or speech. There are two common types of n-gram in computational linguistics, namely, character-based n-gram and word-based n-gram. The application of both types of n-gram is studied [35, 17], tracing back to the early years of research on the text classification.

One difference is that the character based n-gram tolerates the textual errors, which can not be captured by the word-based n-gram. For instance, 'hello world' and 'hellow orld' are treated as two totally different bigrams by the word-based n-gram. But character-based n-gram can capture a lot of character sequences combinations. Clearly, this robustness of the character-based n-gram is achieved by computing more features.

In the context of our task, where privacy policies are often formally drafted by lawyers, fewer than normal contextual errors are expected. Hence, we only apply word-based n-gram in our experiment.

Term extraction Term extraction methods are used to extract term-level features that are specific words and expressions found within the native documents that are generally representative [25]. Term extraction method normally is constructed as a pipeline of tokenization, text normalization, generation of phrases and filtering. Regardless of various methodologies applied, the goal of term extraction is to construct terms, by selecting meaningful terms from bag-of-words and/or constructing

new terms (phrases) from combination of words.

We apply two different term extraction strategies. The first is simple filtering of terms based on predefined POS tags[9]. The second is a full pipeline[10] of term extraction.

4.3.2.3 Concept-level representation

In this subsection, we present two types of concept-level representation, i.e. the concept-level representation based on POS tagging and the one based on WordNet. The goal of creating concept-level representation is to preserve more semantic/syntactic information from the original sentences. Compared with term-level representations, concept-level representation can capture specific semantic information. As shown previously, POS tagging can preserve the parts of speech information which helps to distinguish different forms of the word 'book'. Furthermore, the WordNet based concept-level representation can preserve the semantic that links together two different words with same meaning.

Throughout this subsection, we use 'POS tagged word' and 'concept' interchangeably for concepts created from POS tagging. And 'synset', 'sense' and 'concept' are interchangeable for WordNet based concepts.

POS tagging based concept representations To begin with, we apply a model called *bag-of-POS-tags*. That is, each sentence is processed by a POS tagger and a POS tag is assigned to a word if applicable. Then, the bag-of-words is reduced to a bag-of-POS-tags by simply replacing all the tagged words by their POS tags.

Though very simple, the information carried by POS is useful in many cases. Finn et al. [27] apply bag-of-POS-tags to J48 (a variant of the C4.5 pruned decision tree algorithm) to classify WWW pages into two classes (fact and opinion) by POS statistics obtained from a Brill tagger. Pak et al. [72] demonstrate that different POS have different effects in sentiment analysis.

In sentiment analysis text classification tasks, different parts of speech also contribute differently to particular categorizes [72]. By analyzing this information, the basic grammatical nature of the task is revealed.

[9]For POS tagging, POS tagger applied is the NLTK recommended Maxent Treebank POS tagger trained on Penn Parsed Corpora.

[10]Full term extraction package Topia - http://pypi.python.org/pypi/topia.termextract/.

Besides the bag-of-POS-tags, another way to make use of the POS tagging process is to construct *bag-of-POS-tagged-words*. The tagged words combine the advantages of both bag-of-words and bag-of-tags, by preserving POS grammatical information and statistics of the words.

Bag-of-synsets based on WordNet The concepts generated based on WordNet are more abstract than the ones generated based on POS. For instance, given the lexical contexts, WordNet hypernymy is applied to link two similar nouns together by a same sense both nouns share. In case of the POS based concepts, these two nouns are detected to be nouns, but the link to their same origin sense is not discovered. This reflects the fact that WordNet based concept representation is a further step based on the POS tags. Technically, this is true because finding the right sense requires the knowledge of the part of speech first, given different parts of speech appear in different taxonomies in WordNet.

The application of WordNet hypernym in text classification context is studied in both rule based and machine learning based methods [92, 93]. It is shown that the WordNet hypernym can bring advantages in specific machine learning tasks.

As discussed in previous section 4.2.2.2, the task of choosing the right synset to use is not trivial. We implement both methods introduced before, namely, the first sense heuristic and the internal maximization method. Particularly, we create four variants of the internal maximization method: the first variant selects the sense of a word that maximizes the sum of word similarity with another word in the sentence, meaning that all similarity scores between any possible senses of two words are summed up to find the max one; the second variant selects the sense of a word that yields the max sum of similarity scores with all senses of all words in the sentence; the third and the forth variants are derived from the first and second variants respectively, by applying the same schemes to the corpus of the whole training set.

4.4 Experimental evaluation

In this section, we carry out experimental evaluation of the techniques proposed in previous sections. The goals are:

Goal 1: For machine learning based approach, systematically examine the impact of different options in each feature engineering technique and analyze the

experiment outcomes.

Goal 2: Select the classification models that optimize the performance on the share statement tasks by combining classifier with feature engineering methods.

Goal 3: Evaluate various models on the final test dataset, compare the results of both the machine learning based methods and the sentence similarity measure based methods.

4.4.1 Datasets

In section 4.1, we introduce the tasks of share statement understanding, i.e. to classify an unseen share statement test item into one of the three classes or into one of the eight classes for the 3-class task or the 8-class task respectively. To facilitate our empirical evaluation of different models, we gather share statements from on-line privacy policies and form a dataset as a benchmark for the 3-class task and 8-class task.

The **first challenge** for building the dataset is that the distribution of the share statements on different categories is highly unbalanced. Majority of the share statements fall into few classes and only a few share statements fall into the other classes. This is the case for both the 3-class task and the 8-class task.

As shown in table 4.3, which is an illustration of a random selected dataset contains 95 well structured privacy policy (referred as **95 privacy policy set** from now on), the imbalance of the distribution of share statement is indeed evident. For the 3-class task, out of 64 share statement samples, 54 samples (84.38%) fall into the the neutral category along with only 6 samples (9.37%) and 4 samples (6.25%) fall into the positive and the negative classes respectively. Similarly, for the 8-class task, the top 2 classes, namely the 'Share only under consent' and the 'Share only under consent with exceptions', consist of 64.06% of all samples.

The **second challenge** for building the dataset is that privacy policies, though natural language documents, are formal policy documents that demonstrate strong patterns and fixed formats in expression. This attribute of privacy policy is both bless and curse. Fortunately, only with the presence of this attribute of privacy policy, it is possible to find the patterns in share statement in order to effectively apply machine learning and natural language processing techniques for to understand share statements by classifying them.

However, there are also problems. For building the dataset, limited patterns in

Table 4.3: Statistics of share statement from 95 well structured privacy policy

3-class	8-class	Count	Percentage (%)
Positive		**6**	6.32
	Not sell Not share	4	
	Not sell	2	
Neutral		**54**	56.84
	Share only under consent	7	
	Share only under consent with exceptions	15	
	Share for exceptions	26	
	Not share for marketing purpose	6	
Negative		**4**	4.21
	No limit share	3	
	Sell and share	1	
No[1]		25	26.31
Outliers[2]		6	6.32
Total		95	100.00

1: No share statement is found in the privacy policy.
2: Share statement does not fall into any category.

share statements lead to limited number of diversified samples in each category for a machine learning classifier to learn enough information. This may lead to under-fitting for the classes where not enough diverse samples are provided and further give rise to the chances of overfitting caused by under-fitting.

With these two challenges in mind, we build the dataset, called Share Statement Dataset (SSD). The SSD contains two parts, the training set and the final test set. Table 4.4 and 4.5 demonstrate the categorical statistics of SSD training and test sets.

The SSD training set contains 75 samples in general. We use three sources of privacy policies to build this training dataset:

First, during the process of annotating the *95 privacy policy set* (see table 4.3), the first 29 samples of share statements are used to form the first part of the training.

Second, we try an automatic process in the hope to automatically 'mine' the share statements from full text privacy policies. For this purpose, a one-class SVM [57] is trained on these 29 samples. Using this trained classifier, another 12 samples are formed by searching about 200 full text privacy policies. After some efforts in parameter tuning of the one-class SVM, this approach is proven to be not as promising as expected.

Table 4.4: Share Statement Dataset (SSD) training set

3-class	8-class	Count	Percentage (%)
Positive		**15**	20.00
	Not sell Not share	8	
	Not sell	7	
Neutral		**47**	62.67
	Share only under consent	9	
	Share only under consent with exceptions	12	
	Share for exceptions	20	
	Not share for marketing purpose	6	
Negative		**13**	17.33
	No limit share	6	
	Sell and share	7	
Total		75	100.00

1: No share statement is found in the privacy policy.
2: Share statement does not fall into any category.

At last, we turn to manual collection by surfing and searching in the Internet. The merit of this approach is that we can solve both **challenges** as stated in 4.4.1. By intentionally omitting some of the share statements found during the web surfing that belong to the more-common categories and by intentionally finding diverse patterns and samples for the less-common categories, we collect the last 34 samples of the training set with the best efforts to *normalize the distribution of samples over all categories* and *to avoid the under-fitting of the less-common categories by getting more diversified samples*.

This effort is reflected in figure 4.3. In contrast to the distribution of the *95 privacy policy set*, the *SSD training set* has a much flatter distribution over the categories, especially for the less-common categories, such as 'Not Sell', 'No Limit Share' and 'Sell and Share'. Furthermore, as mentioned above, while collecting these samples, extra efforts are made to find diverse pattern/sentences for these categories.

On the other hand, the SSD test is built to stimulate the natural distribution of the share statements. This means, we want to use the SSD test set to approximate a sample set that a user may encounter during normal period of Internet surfing.

First, we assume the *95 privacy policy set* itself is such an approximation. Though this set contains only well-structured privacy policies, which may not be provided for all the websites a user visits, we believe that the Internet surfing activities of an average user are expected to be limited to the scope of well-known websites

Table 4.5: Share Statement Dataset (SSD) final test set

3-class	8-class	Count	Percentage (%)
Positive		**4**	10.81
	Not sell Not share	3	
	Not sell	1	
Neutral		**31**	83.78
	Share only under consent	3	
	Share only under consent with exceptions	8	
	Share for exceptions	14	
	Not share for marketing purpose	6	
Negative		**2**	5.41
	No limit share	1	
	Sell and share	1	
Total		37	100.00

in majority of the time. And these well-known and frequently-visited websites are mainly operated by major companies and tend to have well-structured privacy policies.

Second, the approximation is now reduced to making sure the *SSD final test set* has a similar categorical distribution as the one of the *95 privacy policy set*. This desired similarity is shown in the figure 4.3.

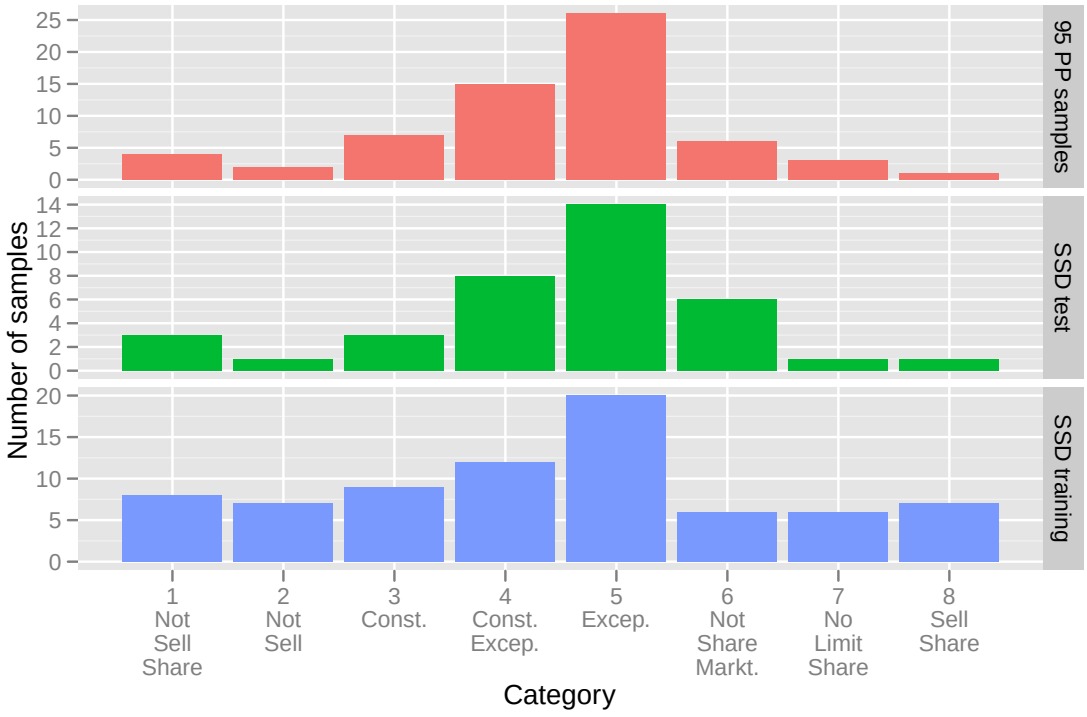
The *SSD final test set* consists of 37 samples. These are share statement samples from the *95 privacy policy set* by excluding the ones used in the first step of building *SSD training set*. There are two exceptions that two samples[11] are created to make sure each category in the test set has at least one sample.

It worths to point out that the *SSD final test set* is not used in any process during the experiments for training and tuning. The classifiers only see this test samples in the final test phase after learning is done.

Evaluation of text classification As already introduced in section 3.3.3, we will mainly apply F_1 and $F_{0.5}$ in the following experimental evaluations. As previously stated in section 4.4.1, our dataset is limited especially for the rare categories, and we only have two separate datasets, i.e. a bigger training set and a smaller final test set, hence we will apply 10×10 cross validation and 50×10 cross validation in different situations.

[11] See Appendix B for details.

Figure 4.3: Categorical distribution of datasets



4.4.2 Single step comparison

To accomplish goal 1 defined in section 4.4, we carry out comparative experiments on high level options such as different algorithms and feature representation methods, and also on low level options such as different feature engineering techniques and even different variants for one feature engineering technique.

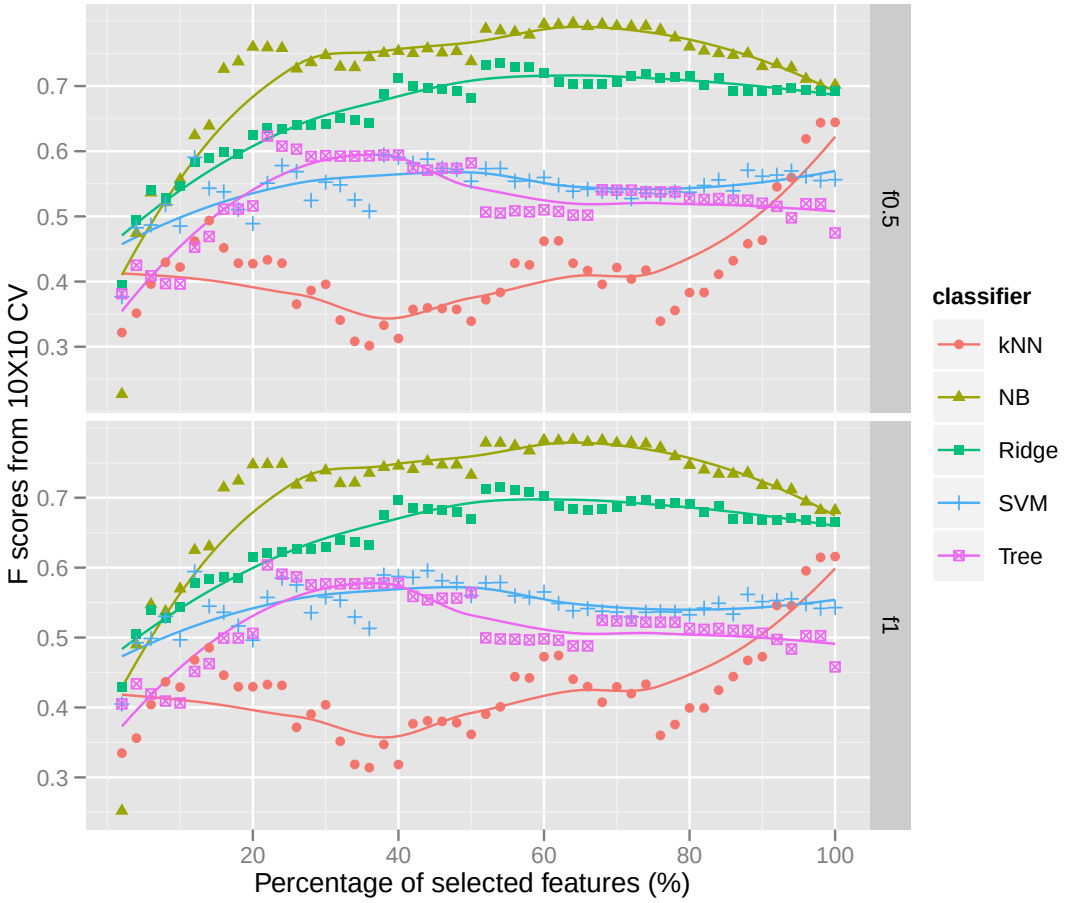
4.4.2.1 Comparison of classifiers

First, we examine the performances of different classification algorithms[12] in a basic setup, i.e. without stopword removal, no stemming, no further feature engineering along with basic bag-of-words representation.

We illustrate the performances of selected classifiers based on different levels of aggressiveness in χ^2 feature selection. In an effort to provide more accurate results, 10×10 fold cross-validation is applied, which means each point in the figures is averaged on the results generated by 10 runs of 10 fold cross-validation.

[12]kNN: $k = 3$, NB: Bernoulli Naive Bayes, SVM: linear SVM with l2 regulation and $C = 1000$, Tree: with $max\ depth = 10$ and $min\ split = 2$.

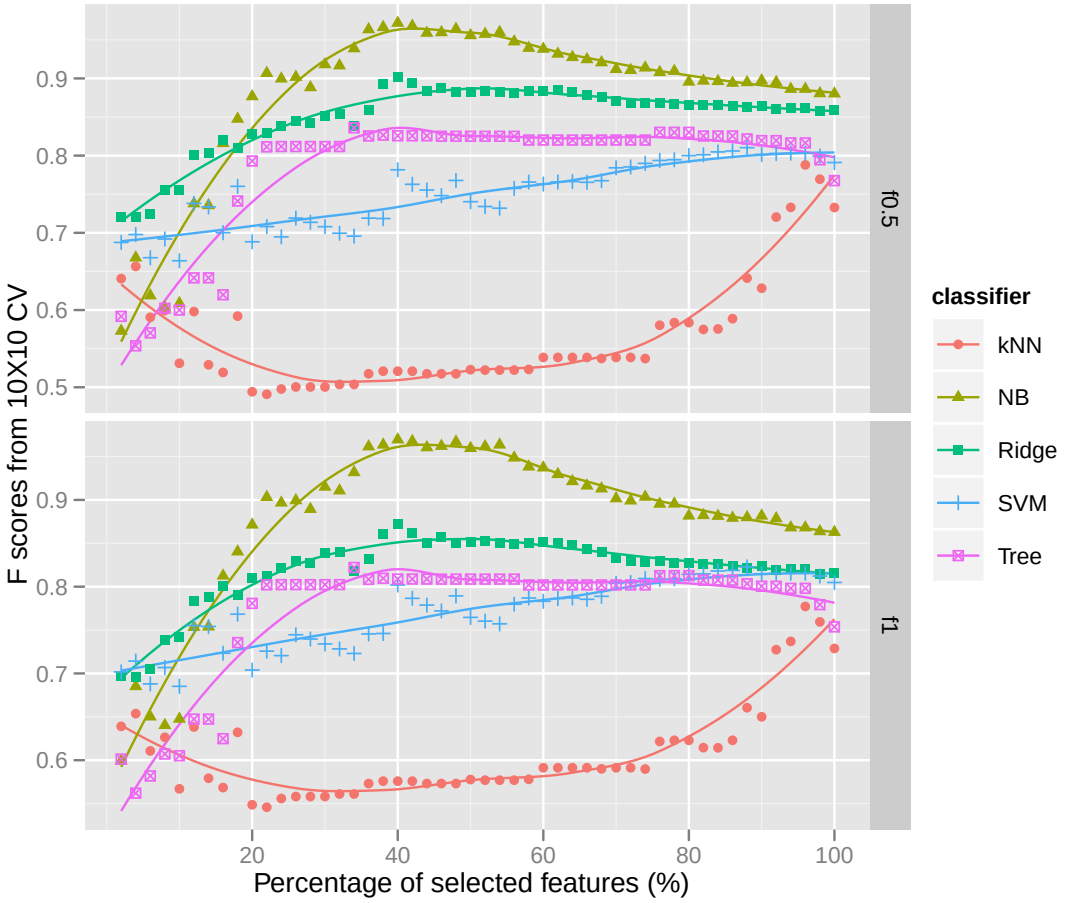
Figure 4.4: Comparison of classifiers for 8-class task in basic setup



For both the 8-class task and the 3-class task, we report the F_1 score together with the $F_{0.5}$ score. For 8-class task, the F_1 score is more important as it is an unbiased reflection on both precision and recall. However, for 3-class task, we believe the $F_{0.5}$ score, which is influenced more by precision, is of greater interests for the reason that it is more critical to prevent assigning positive label to negative samples rather than to assign as many positive labels as possible in case of 3-class task.

As shown in figures 4.4 and 4.5, Naive Bayes and Ridge classifiers show constant superiority over the other three classifiers. The results about Naive Bayes and Ridge classifier also confirm the similar observations in previous topic classification task. Ridge and Naive Bayes classifiers in both tasks reach best performance at the points about half of the features are chosen by χ^2 feature selection. One of the superiority of Ridge classifier is that it constantly performs in the best level, and even starting

Figure 4.5: Comparison of classifiers for 3-class task in basic setup



from very aggressive feature selection phase, where only 2%-10% features are selected. Further, though SVM is not ranked in the top tier, it may because of the specific characters of the two tasks in the basic setup. Due to the similarity between Ridge and SVM, we will return to SVM when we compare the performances using the test set, even though we will not further address SVM under comparison using the training set.

While comparing across the two figures, a straightforward observation about the two tasks is the difference in difficulty. These two tasks, by nature, are text classification tasks based on the same training data but with different categories. Hence, the reason why 3-class task is easier than 8-class task is obvious. Another observation is about the subtle distinction in the shapes of Naive Bayes classifier. This could be a sign that the peak of Naive Bayes' extraordinarily high F_{scores} in the 3-class task may be resulted by overfitting.

Grounded on the results of the empirical analysis of classifiers in this step, we will mostly apply either one or both of the Naive Bayes and Ridge classifiers in our later discussion about feature engineering techniques.

4.4.2.2 Stopword removal and stemming

Stopword removal We test four different options for stopword removal on both Naive Bayes and Ridge classifiers. The experiments in this step are carried out on both the 3-class and 8-class tasks. We test 50 runs of 10-fold cross-validation for each of the four scenarios.

Option 1 - Common English stopword list. One of the most common approaches for stopword removal is using a common English stopword list. We apply this approach as a comparison baseline. The common English stopword list is provided in Appendix C.3.

Option 2 - Threshold stopword removal. Another common way to remove stopwords is using a threshold. We set a threshold to eliminate the top 20 most frequent words.

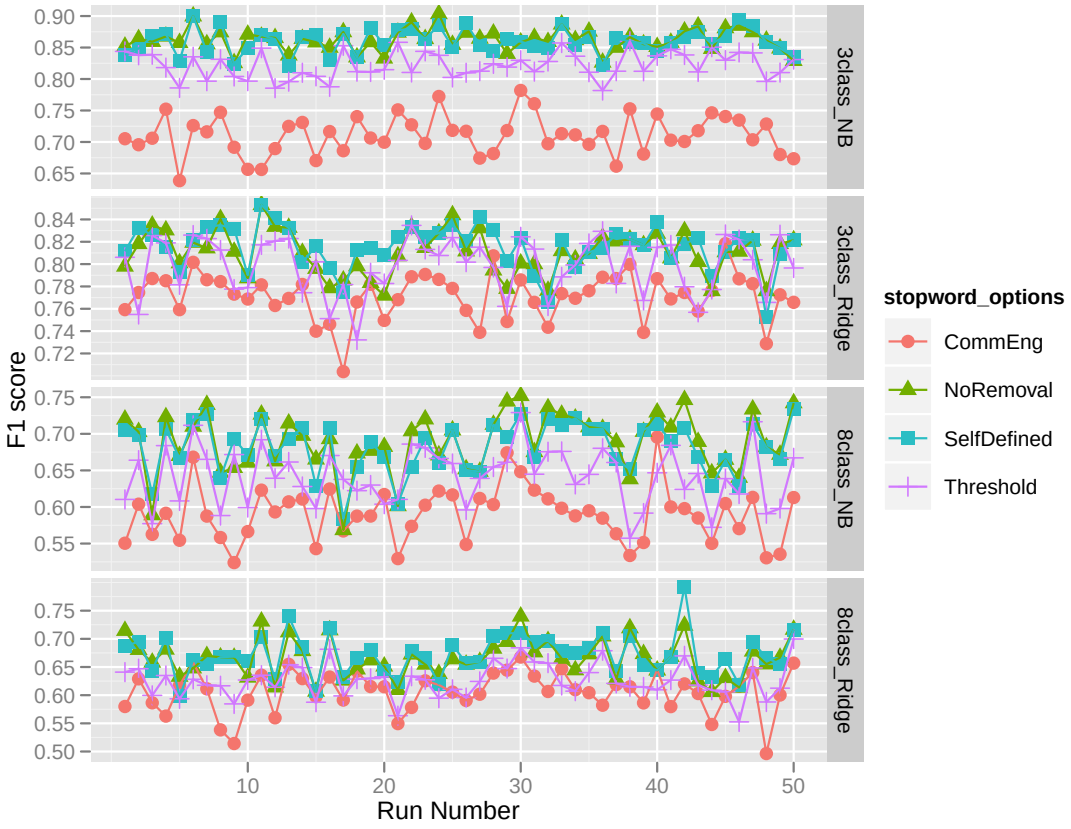
Option 3 - Self defined stopword list. Instead of a common English stopword list, one can define a specific stopword list. In this option, we apply a stopword list simply consists all the company and brand names occurred in the sample privacy policies.

Option 4 - No stopword removal. Same as in the setup of previous step where we compared classifiers, using no stopword removal is another option.

As shown in figure 4.6, the option 3 and 4, self-defined stopword list and no stopword removal, have shown constant better performances than other options while option 1, common English stopword list, is always the worst performer.

The key to interpret such results lies in the special roles played by parts of speech that often removed by common English stopword list. For instance, some common adverbs and conjunctions provide critical information to separate different classes in our two tasks. By eliminating all common English words, the common English stopword list option is not able to preserve this information and therefore shows constant worse performances in all four scenarios. Similarly, the threshold option

Figure 4.6: Comparison of stopword removal options



removes some of these critical common words and hence performs in between of the best performers and the common English stopword option.

The major difference between option 3 and 4 is that the self-defined stopword list removes the specific company and brand names. Despite the similar performances between the two options, the self-defined stopword list option is deemed to be a better option considering its potential ability in preventing overfitting caused by these specific names.

Also the figure 4.6 provides a possible means to compare the general performance change of Ridge and Naive Bayes classifiers in response to feature engineering. Naive Bayes shows more active performance changes given different options in feature engineering.

The difference in variances as shown in the figure also supports the superiority of option 3 and 4 due to their lower variances. From now on, unless otherwise noted, we will apply the no stopword removal for further step-wise comparisons and self-defined stopword removal for final result analysis.

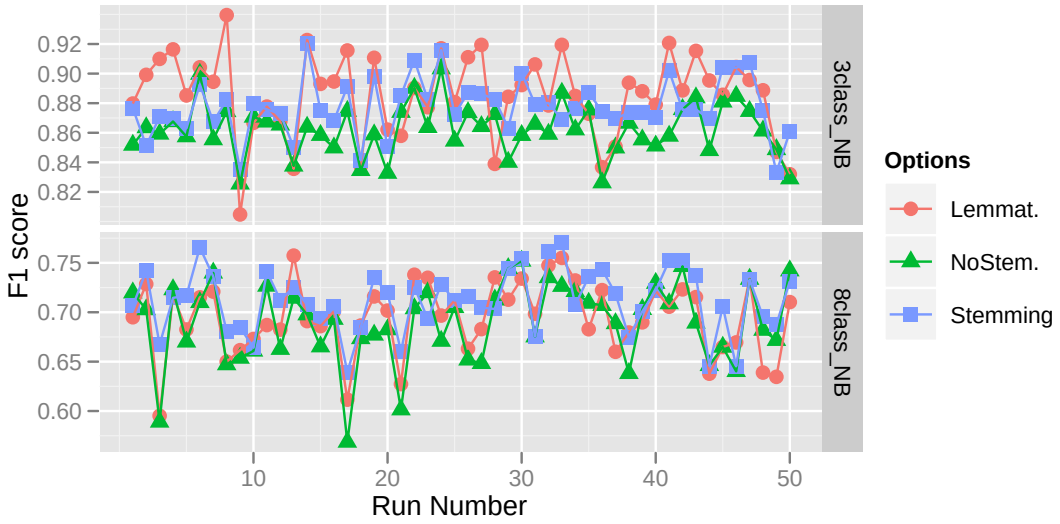
Stemming As previously described in section 4.3.2.1, several options of text normalization can be applied for text classification tasks. In this step, we compare both stemming and lemmatization to the baseline where no stemming is used.

Option 1 - Stemming. We apply the widely used Porter stemmer [79] to reduce inflected words to their stems. We allow the stemmer to handle all words in the sentence automatically.

Option 2 - Lemmatization. The lemmatization process combines the WordNet lemmatizer [66] with an NLTK recommended Maxent Treebank POS tagger. The POS tagger first determines the POS tag of a word, and then the WordNet lemmatizer takes both the POS tag information and the word itself as input for lemmatization. For the words that Maxent Treebank POS tagger gives no output, the word itself is input into the lemmatizer.

Option 3 - No stemming. The baseline for comparison.

Figure 4.7: Comparison of stemming options



As can be observed in figure 4.7, both stemming and lemmatization do not provide substantial boost on the F_1 scores. Considering the extra computational costs of the stemming methods, especially the lemmatization which requires POS tagging and WordNet lookup, the slight increase in performance does not provide strong support for applying the stemming methods. Therefore, we will not apply any stemming methods in the further steps or in the final methods.

4.4.2.3 Term based representations

n-gram n-gram helps to capture the phrases that consists n words. We test three options of n-gram and compare them with unigram which is the baseline in this step and equals to the baseline used in previous step.

Option 1 - Bigram. Bigram preserves both the single words and possible phrase combinations of two words. In the experiment, we simply count in all possible two-word phrase and then apply χ^2 feature selection to select same amount of features as in the baseline.

Option 2 - Negation bigram construction. In this option, we apply the negation bigram construction method as proposed in section 4.3.2.2. Specifically, we implement two negation construction patterns which are concluded from observations on the training dataset:

$P_1 = (\ \{\text{'}not\text{'}, \text{'}never\text{'}, \text{'}neither\text{'}\},\ 10,\ \{V\})$

$P_2 = (\ \{\text{'}without\text{'}\},\ 3,\ \{N\})$

Same as in option 1, the number of features from negation bigram construction are reduced by χ^2 to the same number of unigram.

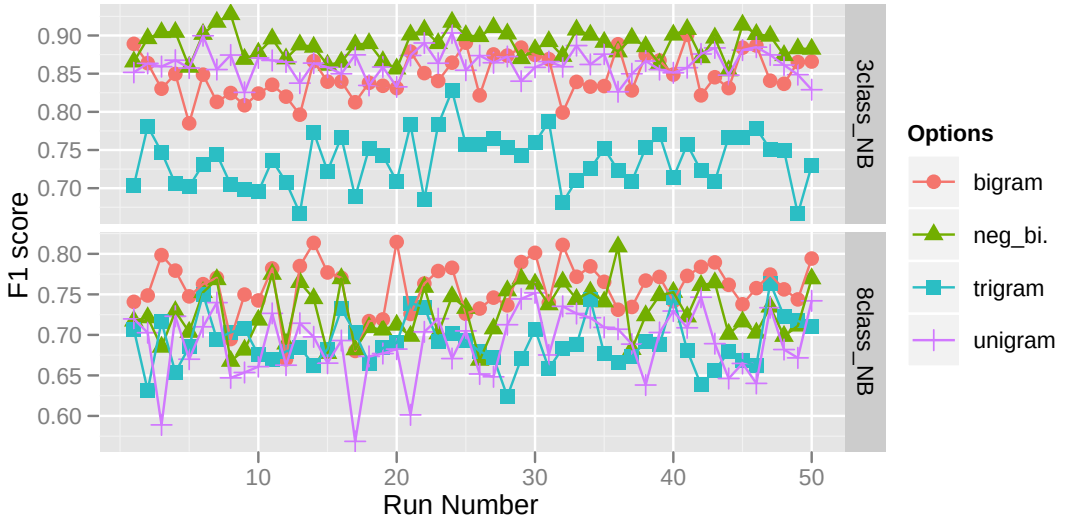
Option 3 - Trigram. Trigram also keeps the three-word phrases along with all features generated by bigram. The feature selection is same as in bigram.

Option 4 - Unigram. The baseline for comparison.

Figure 4.8 shows the F_1 scores of the options for n-gram. In the 3-class task, negation bigram construction performs better than the baseline, unigram, which performs slightly better than bigram, while the trigram has significantly worse performance. In the 8-class task, the performances of the four options are close with slight priority shown by bigram and negation bigram construction.

Demonstrated in the results, negation bigram construction and bigram can both improve the performance measured by F_1 score. However, we prefer the negation bigram construction to the traditional bigram method for the reason that bigram may lead to overfitting by retaining certain non-generalized two-word phrases meanwhile our method of negation bigram construction specifically targets to useful negation bigrams with justified semantic meanings.

Figure 4.8: Comparison of n-gram options



Term extraction In contrast to n-gram, term extraction methods do not create extra features but actually reduce the number of features by extracting only the useful terms out of all word features. However, because the very limited amount of features in the share statement tasks, we expect term extraction will not improve or may even deteriorate the performance. We examine two options of term extraction to test this expectation.

Option 1 - Term extraction by POS. A Maxent Treebank POS tagger from NLTK package is applied to tag all the words. Only the words in specific POS tags, which normally posses real meaning, are extracted. In the experiment, only words tagged with POS tags that begins with 'V', 'N', 'R' or 'J' are extracted.

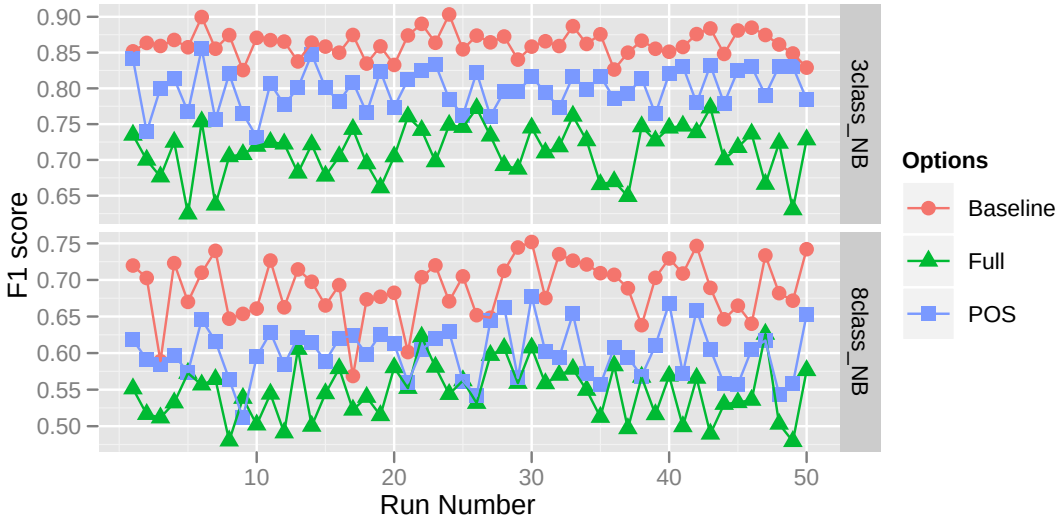
Option 2 - Full pipeline term extraction. A full package[13] of term extraction is applied. It is more aggressive than the option 1 and mostly only preserves the nouns and nouns phrases.

Option 3 - No term extraction. The baseline for comparison, same as the ones used in previous two steps.

As proven in figure 4.9, the two term extraction options indeed do not improve F_1 performance but considerably deteriorate the F_1 score. It is also clearly noticeable that the more aggressive term extraction strategy leads to worse deterioration

[13]Topia, as noted previously in section 4.3.2.2.

Figure 4.9: Comparison of term extraction options



of the performance. Hence, we will not apply any term extraction method in our further steps or final result analysis.

4.4.2.4 Concept based representations

POS based representation We test the POS based representation first. The main aim in this step is to compare the POS tagged words model to the baselines which are described in option 2 and 3 below.

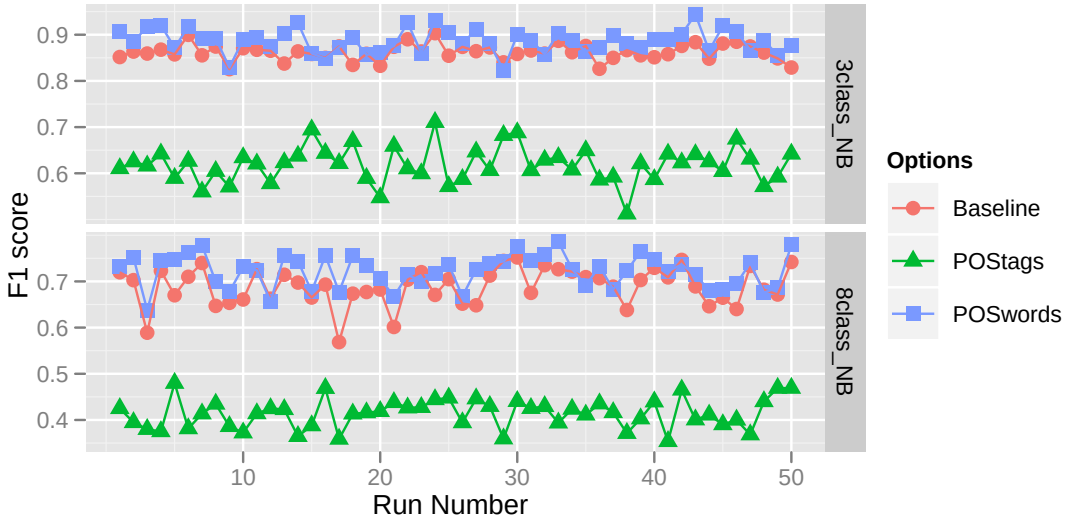
Option 1 - Bag-of-POS-tagged-words. This is the core POS based representation. It adds potentially useful semantic information to the original words and thus forms features that go beyond the bag-of-words model.

Option 2 - Bag-of-POS-tags. Bag of POS tags model only keeps the basic POS tag information of the sentences. This shows the pure capability in distinguishing different categories provided by the simple semantic information from POS tags. We do not expect any good performance from this option because it is rather a baseline of semantic information.

Option 3 - Baseline. The baseline for comparison, same as the ones used in previous steps. It is the baseline using traditional bag-of-words model.

Option 2 shows surprisingly reasonable F_1 performance even only using the POS tags. Particularly, about 0.6 and 0.4 F_1 scores are achieved for the 3-class

Figure 4.10: Performance of POS based representation



and 8-class tasks respectively. This justifies the basis of this step that semantic information carried by POS tags is substantially useful even without the word statistics conveyed by the bag-of-words models.

The POS based representation tested as option 1 demonstrates constant better performance than the baseline option. As will be shown in later section 4.4.2.5, the option 1 of POS based representation holds the best performance among all single step feature engineering methods.

In addition to its improvement in performance, the experiments of POS based representation also confirms semantic information's positive effects in short text classification tasks. It underpins the rationale that semantic information helps to cover the useful features that lead to further improvement in classification performance but are left out by bag-of-words model.

WordNet synset based representation WordNet synset models are considered as another concept based representation. By generalizing words into sets of synsets, synset based representation provides general definitions for sets of similar words which share the same hyponymy and records the various semantic relations between these synonym sets.

However, as already discussed in preceding section 4.2.2, constructing good synset based representation requires solution for WSD problem. In other words, the performance of a synset based representation hinges upon the quality in handling

the sense disambiguation task. First, we compare 5 different WSD options with the baseline. This experiment aims to show whether synset based representation can boost the performance in general.

Option 1 - First sense heuristic. Synset based representation is constructed by simply selects the first sense of each word which has multiple senses.

Option 2 - Sentence internal word maximization. One variant of the internal maximization method that selects senses maximizing similarity with another sense in the scope of the sentence. In our implementation, we apply brute-force comparison that search all possible combinations to decide the maximums.

Option 3 - Sentence internal maximization by sentence sum. One variant of the internal maximization method that selects senses maximizing similarity with all other words in the scope of the sentence. Brute-force search is implemented as well.

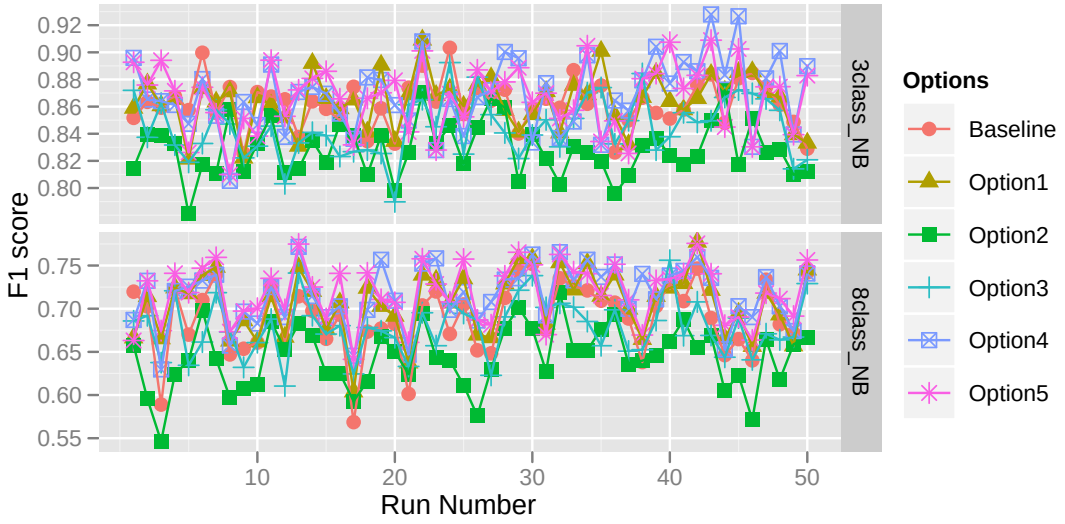
Option 4 - Corpus level word maximization. One variant of the internal maximization method that selects senses maximizing similarity with another sense in the scope of the whole training corpus. Brute-force search is implemented as well. Due to the large amounts of comparisons computed within the corpus scope, the computational cost is considerably high in this option.

Option 5 - Corpus level maximization by sentence sum. One variant of the internal maximization method that selects senses maximizing similarity with all other words in the scope of the whole training corpus. Brute-force search is implemented as well. The computational cost is similar to option 4.

Option 6 - Baseline. The baseline for comparison, same as the ones used in preceding steps.

Various WSD options yield slightly different performances as can be observed in figure 4.11. It is evident that better WSD scheme leads to slightly better F_1 score. Among the four variants of the internal maximization method, corpus based options, option 4 and 5, outperform the other two options, which are sentence based. We expect such results for the reason that option 4 and 5 are generally better WSD schemes compared with option 2 and 3 considering that whole corpus

Figure 4.11: Comparison of WSD options for synset representation



is used in the process of sense disambiguation in option 4 and 5. Searching for the sense that maximizing similarity score in the whole corpus grants a better chance of find the right sense than searching only in the sentence. The difference between the two sentence based options, namely option 2 and 3, can be explained by inter-word similarity maximization's relative inferiority to sentence internal similarity maximization scheme.

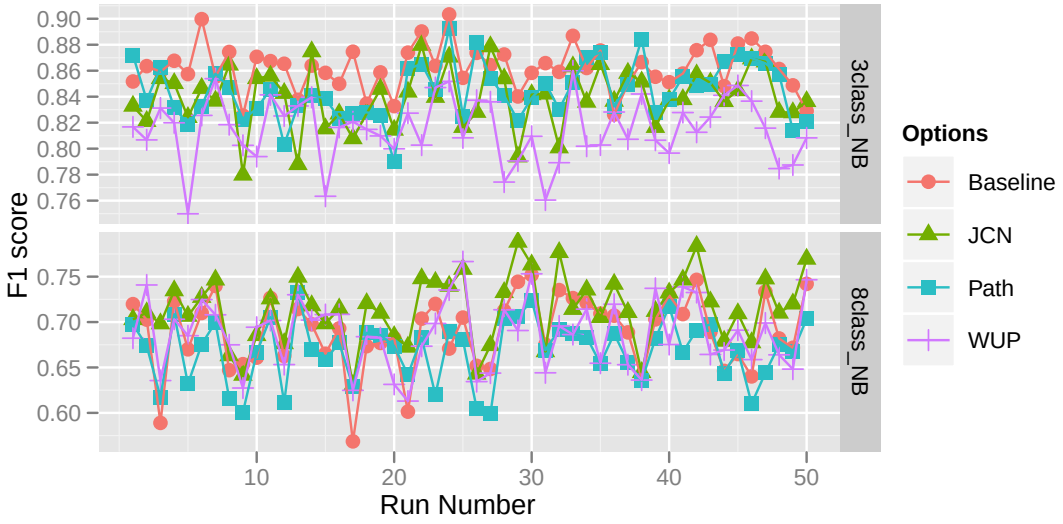
In general, option 1, 3, 4 and 5 all demonstrate somewhat better F_1 performance than the baseline. However, for the similarity maximization based options, i.e. option 3, 4 and 5, the slight improvement in performance can hardly justify the high computational costs. Regarding to the first sense heuristic option, the option 1, however, the slight boost is favorable because first sense heuristic is much cheaper computation-wise.

In previous experiment where WSD options for synset representation are tested, we applied the path distance similarity measure for all options. However, as introduced in previous section 4.2.2.1, there are various methods for calculating the WordNet synset similarity. To justify the conclusion in previous step, we will quickly examine three similarity measure, each represents a specific scheme.

Option 1 - Path distance similarity. The simplest WordNet based similarity measure.

Option 2 - Wu-Palmer (WUP) similarity. Depth based similarity measure that

Figure 4.12: Comparison of similarity options for synset representation



also takes the least common subsumer into consideration. We apply this measure as a representative of the depth based measures.

Option 3 - Jiang-Conrath (JCN) Similarity. Representative of the information content based measures. All option 1, 2 and 3 are described in section 4.2.2.1.

Option 4 - Baseline. The baseline for comparison, same as the ones used in preceding steps.

As demonstrated in figure 4.12, we can arrive at a general conclusion that the choice of similarity measure does not significantly effect the performance. Each method has similar performance compared with the baseline and path distance similarity. An exception is the JCN method slightly outperform other three options in the 3-class task. However, because JCN similarity measure requires to calculate the information content in each run, there are extra computational cost on top of the basic computational cost for each WSD option. Therefore, we will stick to the basic path distance similarity measure which is already effective enough to judge the WSD options.

4.4.2.5 Summary of single step feature engineering methods

To sum up all the single step feature engineering methods, we present the table below to list all the methods with their detailed information.

Table 4.6: Summary of single step feature engineering

Step	**Option**	**Feature (%)**[1]	**Time (s)**[2]	**3-class task**[3]		**8-class task**[3]		Δ^4
				F_1	$F_{0.5}$	F_1	$F_{0.5}$	
Baseline		100.00	0	86.24	88.13	68.96	71.13	0.00
Stopword	Comm. Eng.	73.65	0	71.15	74.53	59.05	60.65	-12.27
	Self Def.	90.25	0	86.02	87.74	68.22	70.14	-0.58
	Threshold	93.50	0	82.24	84.24	64.26	66.05	-4.42
Stemming	Stemming	85.92	0.07	87.78	88.93	71.20	72.84	1.57
	Lemmat.	90.61	12.4	**88.57**	**89.98**	69.49	71.53	1.28
n-gram	Bigram	354.15	0.03	84.75	83.98	**75.90**	**76.65**	1.71
	Neg. Bi.	109.75	10.1	**89.48**	**91.23**	**72.87**	**74.44**	**3.39**
	Trigram	684.84	0.06	73.71	70.96	69.05	69.57	-7.79
Term ext.	POS	82.31	10.2	79.99	82.94	60.12	62.02	-7.35
	Full	55.60	0.3	71.29	73.82	54.74	56.18	-14.61
Pos rep.	POS tags	9.03	10.3	61.93	59.49	41.52	42.80	-27.18
	POS words	113.72	10.1	**88.91**	**90.87**	**72.22**	**74.05**	**2.90**
Synset WSD	Option 1[5]	95.31	4.6	86.40	87.98	70.65	72.80	0.84
	Option 2	98.92	140	82.85	84.96	64.41	66.19	-4.01
	Option 3	104.33	136	86.23	88.08	67.91	70.25	-0.50
	Option 4	93.14	1721	86.88	88.08	71.59	73.52	1.40
	Option 5	93.86	1723	86.72	87.58	71.86	73.97	1.42
Synset sim	Path	104.33	136	86.23	88.08	67.91	70.25	-0.50
	WUP	107.22	163	85.39	87.53	70.25	72.19	0.23
	JCN	96.75	555	83.96	85.98	71.48	73.56	0.13

1: Number of features, percentage in contrast to the baseline.
2: Computational time cost for preprocessing on the training dataset.
3: F scores for both 3-class and 8-class tasks are shown in percentage.
4: Improvement from baseline, averaged on four F scores.
5: Names for WSD options can be found in previous page.

Size of feature set As noted in the third column of table 4.6, some feature engineering methods increase the size of the feature set while the others lead to reduction of the feature size. In general, a change in feature number is caused by the nature of the feature engineering method. For instance, stemming reduced the feature size because words with same root are reduced to one feature, or bigram and trigram enlarge the feature size because bigrams and trigrams are formed using combinations of single words. Same as in previous single step experiments, all the results shown in the table are measured on the feature set with same size as the baseline. This means, in order to form a standardized comparison, χ^2 is applied to select features in case the feature size generated by a particular method is bigger than the baseline's feature size.

Computation time cost The fourth column demonstrates the computational cost of different feature engineering methods. As the computational costs are measured using computation time on the training set, it is important to specify the details of the experiment environment. All the experiments are carried out on a Windows Vista PC (Intel Core3 Duo T6570 with 2G RAM) using Python 2.7.

There are five methods that have similar computation time costs as they all use the NLTK Maxent Treebank POS tagger, which is a pre-trained tagger that appears to be an NLTK ClassifierBasedTagger trained on the treebank corpus using an NLTK MaxentClassifier. Because the POS representation method is one of the best performers that we will carry on in final result analysis, it is important to point that the NLTK Maxent Treebank POS's computation time cost is considerably higher than other types of POS tagger, such as the Brill tagger [11]. As shown in [76], though the Maxent tagger provides better accuracy in general, non-classifier based tagger has much better efficiency. For example, the Brill tagger can be faster than classifier based tagger by two to three magnitude orders. Therefore, the efficiency of POS taggger related methods will not be a problem in future implementation. If speed is an issue, other types of tagger can be applied with no substantial reduction in accuracy.

For synset related methods, except the first sense heuristic, all other methods and variants require the calculation of the maximization by iterating over a large number of combinations of words and computing their similarities. Such process requires calculating multi-nested loops that are computational intensive in nature. Using current brute-force search implementation, the computational cost is significant and hence the efficiency of this type of methods becomes unjustified.

F scores performance The right half of table 4.6 provides the F scores for each feature engineering method in detail. The F scores are measured in a consistent way, same as applied in previous experiments, i.e. mean average of scores from 50 runs of 10-fold cross-validation.

Marked in bold are the top scores in each item. Two methods, the negation bigram construction, as proposed previously, and POS representation demonstrate continuing better F scores than other methods. Such results have already been discussed in preceding step-wise experiment section. It is worth to point out that the right-most column is the averaged improvement of the four scores for each item. It serves as a clear indication on the perform of a specific method in contract

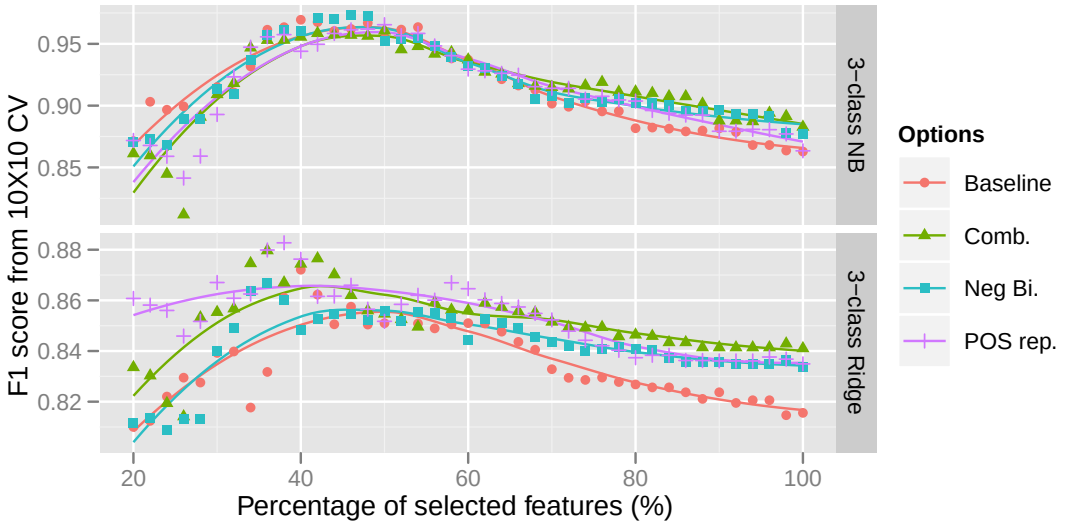
to the baseline method.

To sum up, from an overall point of view, the negation bigram construction and POS representation indeed have beneficial contributions to the improvement of the classification model and they outperform other feature engineering methods in the aspect of F score performance and/or efficiency.

4.4.3 Combining feature engineering methods

To achieve the goal 2 defined in section 4.4, we extend the experiment and discussion of the previous section, by stacking feature engineering methods to form a combined classification model. Particularly, we are interested in examining the classification model formed by combining the negation bigram construction and the POS representation which are two top feature engineering methods shown in empirical comparison results.

Figure 4.13: Combined method vs. other methods (3 class)



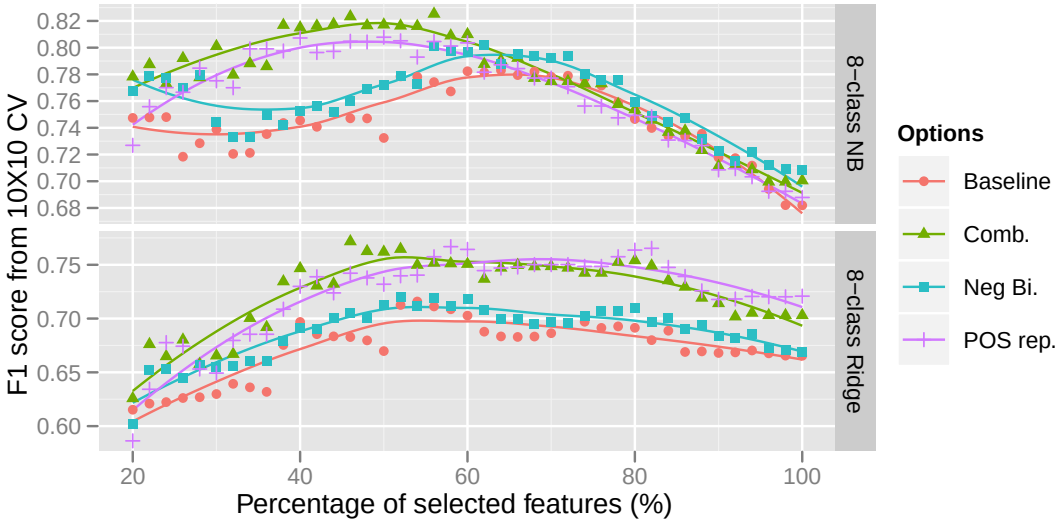
Firstly, if we revisit the implementation of both the negation bigram construction and the POS representation, it is clear that they are different kinds of feature engineering method in that negation bigram construction only adds negation bigram phrase as extra features to the original representation while the POS representation is a whole replacement for the traditional bag-of-words model. Hence, we can easily stack negation bigram construction on top of POS representation. That is to say, the combined feature engineering method is implemented by running both

methods and then combined the outputs from both into a single set of features.

As already shown in previous single step comparison, both methods give rise to steady improvements in F scores. However, this does not automatically lead to a further boost by combining the two methods. Though we expect the improvements are indeed somehow additive, such property is still target for further empirical experiments.

In figures 4.13 and 4.14, we compare the combined method and the two separate methods with the baseline. In the 3-class task, the combined method shows more steady performance throughout all levels of aggressiveness of the feature selection, especially in the tail part where the F scores drop. In the 8-class, the advantages of the combined method is shown more obviously. Using Ridge, combined method constantly shows top performance, while using Naive Bayes, the combined method also keeps the superiority until after the peak and drops similarly as other methods.

Figure 4.14: Combined method vs. other methods (8 class)



For consistency, we present the table 4.7 as a continuity of the table 4.6. Though the combined method's obviously higher F scores in the table may be caused partially by its advantage in feature selection due to its slightly larger feature size measure, it is still evident, according to the information conveyed in both table 4.7 and figures 4.13 and 4.14, that the combined method shows it predominance no matter measured by fixed percentage or fixed number of features.

All in all, the combined method is indeed a better choice than the other two

Table 4.7: Comparison of combined method and other methods

Option	Feature (%)	Time (s)	3-class task F_1	3-class task $F_{0.5}$	8-class task F_1	8-class task $F_{0.5}$	Δ
Baseline	100.00	0	86.24	88.13	68.96	71.13	0.00
Neg. Bi.	109.75	10.1	89.48	91.23	72.87	74.44	3.39
POS words	113.72	10.1	88.91	90.87	72.22	74.05	2.90
Combined	123.83	9.9	**90.77**	**92.38**	**75.58**	**77.07**	**5.33**

single step feature engineering methods which are already selected as the best ones out of many single step feature engineering methods. Though, it may not be able to boost the results always by a sum of the improvements achieved by the two single step methods, it demonstrates better performance and/or more steady performance in various scenarios. It is the final feature engineering model that we conclude as the best choice for our share statement classification tasks.

4.4.4 Final test

Up to this point, we have already selected the feature engineering models for the machine learning based approach and introduced different sentence-level similarity measure based methods. In order to compare the two types of approaches and to verify our earlier conclusions and expectations, in this section, we will test all these methods on the test dataset, which was introduced in section 4.4.1.

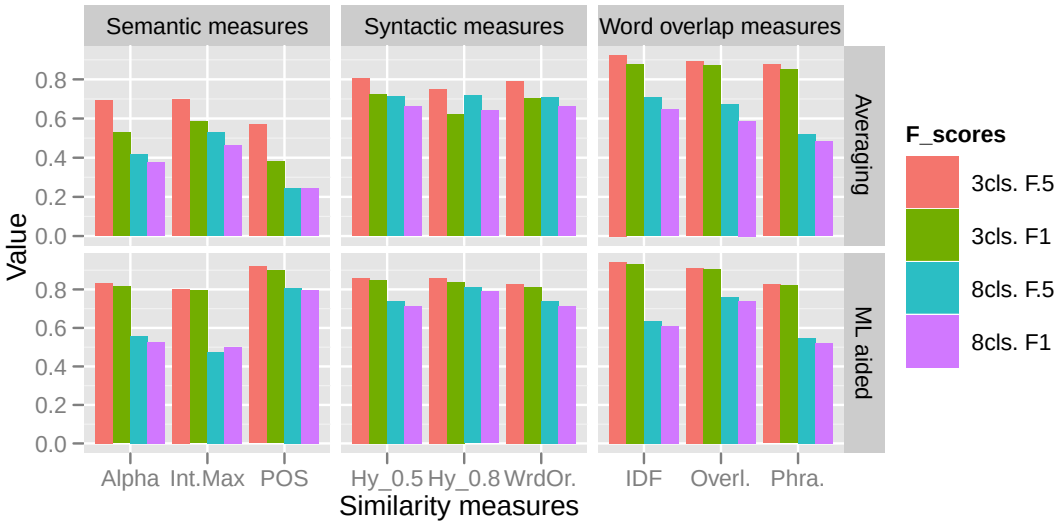
Similarity based approach First, we consider the similarity based approach. As shown in figure 4.15[14] we compare the all the 3 main types of the similarity measures introduced previously. Each of these methods has been tested using both the simple averaging method and the machine learning aided method for classification.

To point out the implementation specifics, there are nine different measures or combinations of measures that are tested using two methods for classification. For the semantic measures, the Jiang-Conrath Similarity method and Brown Informa-

[14]The names of the measures: Alpha - Alpha variant of first sense heuristic based semantic similarity measure; Int. Max - inter-sentence maximization based semantic similarity measure; POS - POS variant of first sense heuristic based semantic similarity measure; Hy_0.5 - hybrid measure where POS and word order measures are weighed equally; Hy_0.8 - hybrid measure where POS semantic measure and word order measures are weighted by 0.8 and 0.2; WrdOr. - word order syntactic similarity measure; IDF - IDF overlap measure; Overl. - simple word overlap measure; Phra. - phrasal overlap measure.

tion Content are applied. For the Alpha variant of the semantic measure, $\alpha = 0.05$ is used. The two hybrid methods, as introduced in section 4.2.3, are combinations of the POS variant of the semantic measure and the word order measure.

Figure 4.15: Test results of similarity based methods

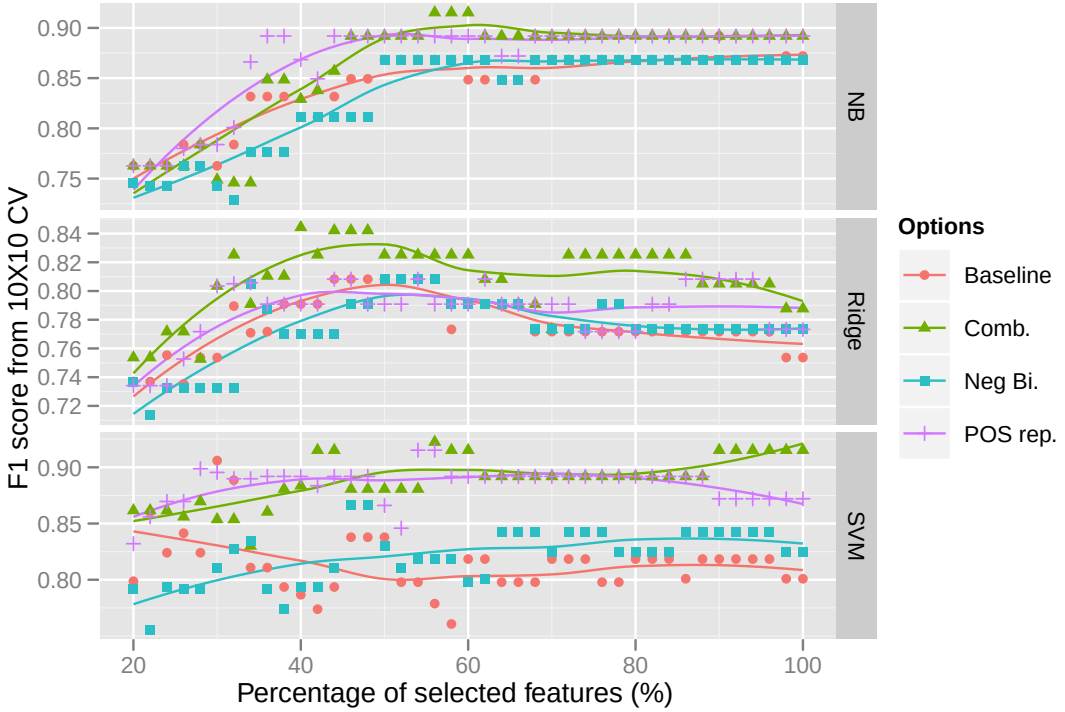


Observed from the first row of the figure, the word overlap measures work surprisingly well given their simplicity and very low computational cost. On the contrary, the semantic measures perform poorly. However, after combined with the word order measure, the POS variant of semantic measure leads to the two hybrid measures that have considerably well performance.

The difference between the first row and the second row for each measure shows the improvement caused by the machine learning aided method for classification. This improvement brought by the machine learning aided classification method is considerably obvious for the semantic measure, especially for the POS variant which becomes the best performer among all the measures. However, the performances of word overlap based measures on the 8-class task are not improved.

Machine learning based approach Second, we test the machine learning based methods on the test dataset. Same as in section 4.4.3, we apply four feature engineering models, namely, the two single step methods, the combined method and the baseline. In this part, we test three classifiers, i.e. Bernoulli variant of Naive Bayes, Ridge and SVM.

Figure 4.16: Test results of selected machine learning based methods (3-class)

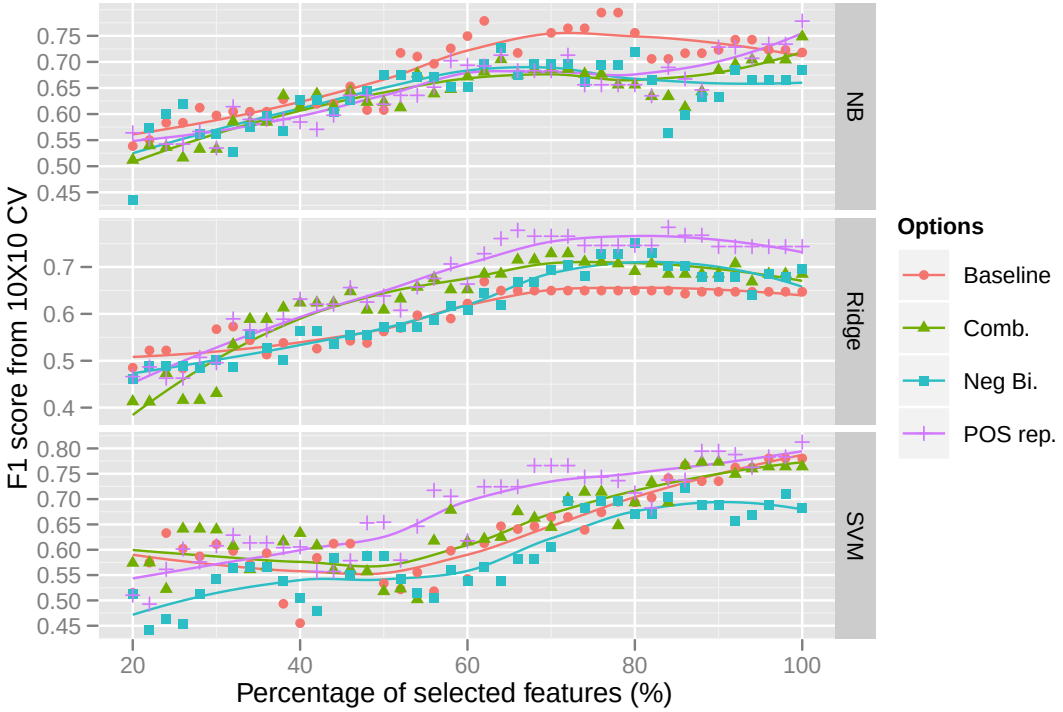


The test results on the 3-class task clearly demonstrates the improvements brought by feature engineering methods in contrast to the baseline, especially, the combined method accomplishes the best results among the four methods with all three classifiers. This improvement varies for different classifiers. For Ridge and Naive Bayes classifiers, the improvement is about 5% in general, which corresponds to the improvement occurred during the training evaluations. And for SVM, this improvement is even higher and up to about 10%.

Same as in previous training phase experiments, Naive Bayes still achieves better performance than the Ridge classifier. However, the difference between the training and the test results with regard to classifiers is the advancement of the SVM. With the combined feature engineering method, it shows top tier performance and achieves even slightly better results than Naive Bayes. One explanation of the SVM's improvement could be the fact that SVM's kernel method is better at exploring the linear separability of the task in higher dimensional space given the limited amount of features provided by the training set for the test set.

For the 8-class task, POS based representation still shows its beneficial contri-

Figure 4.17: Test results of selected machine learning based methods (8-class)



butions while the negation bigram construction performs slightly worse than the baseline. This leads the performances of the combined methods that are better than the baseline for Ridge and SVM but slightly worse for the Naive Bayes.

The SVM shows promising performances again in the 8-class task where actually it provides slightly but distinguishably better performance than Naive Bayes. Ridge, on the other hand, still brings inferior performance among the three classifiers.

Comparison of the two approaches To clearly show the difference between the two approaches for comparison purpose, we list the test results from different combinations of methods and classifiers as shown below. A whole version of the table is available in appendix C.2.

In general, as shown in table 4.8, different methods in the two different types of approach have varied F score performance. It is clear that, when using the simple averaging method for classification, similarity based approach has obvious inferiority to the machine learning based approach. However, when the machine learning aided method is used for classification, the similarity based approach performs

Table 4.8: Comparison of two share statement classification approaches

Type	**Classification / Clssifer**	**Method**	**3-class task**		**8-class task**	
			F_1	$F_{0.5}$	F_1	$F_{0.5}$
Similarity based	Averaging	Overl.	87.05	89.14	58.76	67.14
		IDF	87.83	92.44	64.65	70.54
		Alpha	52.77	69.13	37.73	41.83
		POS	37.94	57.12	24.17	24.09
		WrdOr.	70.40	79.14	65.95	70.58
		Hy_0.8	62.19	74.66	64.01	71.95
		Hy_0.5	72.47	80.48	66.33	71.27
	ML aided	Overl.	90.43	91.19	73.90	76.27
		IDF	**93.22**	**94.04**	60.81	63.53
		Alpha	81.49	83.48	52.89	55.81
		POS	89.81	91.93	**79.76**	80.61
		WrdOr.	81.23	83.03	71.48	74.05
		Hy_0.8	83.93	85.74	78.90	**80.97**
		Hy_0.5	85.02	85.78	71.54	74.16
Machine Learning based	Naive Bayes	Baseline	89.19	89.19	79.45	78.51
		Neg. Bi.	86.85	87.06	72.58	74.03
		POS words	89.19	89.19	77.82	77.49
		Combined	91.52	91.31	74.84	74.35
	Ridge	Baseline	80.83	84.27	64.71	68.34
		Neg. Bi.	80.83	84.27	73.08	74.88
		POS words	80.83	84.27	78.46	80.23
		Combined	84.43	88.54	72.88	74.54
	SVM	Baseline	90.60	91.47	78.04	79.53
		Neg. Bi.	86.61	86.69	72.15	73.33
		POS words	91.52	91.31	**81.26**	**81.37**
		Combined	**92.25**	**92.46**	77.34	78.37

on the same level as the machine learning based approach. More specifically, the machine learning approach has better performances for the 8-class task while the two types of approaches have similar performances for the 3-class task.

When applying Naive Bayes or SVM classifier with the combined feature engineering method or only the POS representation, the machine learning based approach has constantly promising performance for both the 3-class and the 8-class tasks. For the similarity based approach, gaining good performances for both the 3-class and 8-class is more demanding. Using either the POS variant of the semantic similarity measure or the hybrid measure leads to good performances for both the two tasks.

In general, it is clear that both types of approaches can provide certain high level

F score performances, particularly, up to $90\%+$ and $80\%+$ for 3-class and 8-class tasks. However, the machine learning based approach is more favorable than the similarity based approach due to three reasons. Firstly, the performances between 3-class and 8-class are not balanced for many similarity based methods. For instance, the IDF method has the highest 3-class performance among all possible options, however, its 8-class performance is rather poor and worse than most of the machine learning based methods. Secondly, the computational cost of the similarity based approach is considerably higher than the machine learning based approach. Thirdly, the machine learning based methods can provide better generalization. This type of methods is tunned and selected during a thorough training phase and the very similar results in test set further prove the generalization. And machine learning based approach can use classifiers that provide regularization for the purpose of improving generalization.

4.5 Summary

In this chapter, we present solutions to the share statement understanding task which is the subject of our research goal. This task is a further study based on the topic classification from previous chapter. We still apply the generic machine learning approach to the solutions that take semantic information into account and categorize share statements into the pre-defined categories. To this end, we propose, build and experimental evaluate two types of approaches, namely, the similarity based approach and the machine learning based approach. For both approaches, different methods are systematically studied and evaluated during thorough experiments. The experimental results demonstrate that both approaches can lead to quite promising performance for the task of share statement understanding.

Chapter 5

Conclusion

In this work we have answered the research questions which were stated in the introduction. We have presented a framework for automatic privacy policy evaluation using a new machine learning approach. We have studied the applications of text classification techniques in privacy policy paragraph categorization, privacy policy grading and share statement understanding. In this chapter, we formulate the contributions to this research, summarize the implications of our work, and pose some recommendations for future work.

5.1 Contributions and implications

The first main contribution of our work is the privacy policy paragraph categorization scheme, in which we apply text classification to categorize privacy policy paragraphs into pre-defined privacy policy topics. This lays the groundwork for privacy policy grading and visualization. We optimize the classification models throughout intensive experiments and achieve promising results on both the validation and testing datasets. This proves the effectiveness of the machine learning approach in automatic privacy policy evaluation.

The second main contribution is the share statement understanding scheme, using which we can classify share statements into pre-defined categories to automatically understand the implications of the share statements. We propose two approaches to facilitate this scheme, namely, the similarity based approach and machine learning based approach with enhanced feature engineering. We test and select the best variants of both approaches and gain reasonable results on both of the two sub-tasks of share statement understanding. This part of the work

demonstrates the potential of applying machine learning to further tasks in privacy policy evaluation beyond the basic job of classifying privacy policy paragraphs into different topics.

Another contribution of this work is the proposal of the overall framework as well as all the research facilities built in this framework. We implement the search result extraction, privacy policy detection, and text extraction programs in order to detect and extract privacy policy specific information from research results into clear text. We collect considerable amounts of privacy policies and manually label the items to form the datasets for experimental evaluations. All these research facilities serve not only for the proposed framework, but also for any future work in this direction.

To the best of our knowledge, our research is the first attempt to apply text classification in automatic privacy policy evaluation. To be more specific, not only the general schemes of applying text classification in privacy policy evaluation are treated thoroughly for the first time, but also the detailed approaches and methods are original explorations in this new area.

We believe that our research has shown and supported a generic approach to apply machine learning techniques to build privacy policy evaluation systems that are effective, automated, flexible and user-focused.

Further implication We envision this new work as a potential game-changer for the current eco-system of privacy policy. Existing privacy policy presentation alternatives and evaluation methods heavily rely on the adoption and co-operation from the websites. This means that this type of methods has to meet the website's needs and compromise with the websites in order to be widely adopted, assuming no legal requirement will change the landscape in near future.

However, the machine learning approach in privacy policy evaluation makes it possible to present the information from privacy policies in a totally different manner – a user-focused manner. Privacy policies paragraphs are categorized into different topics. Hence, the users can easily browse through only the parts of the privacy policy that they are interested in. This helps to solve the readability issue of privacy policies. Furthermore, a general grade is provided for each privacy policy. The users can quickly get the insights about the overall quality of the privacy policies. This functionality helps to improve user's awareness about privacy and increase the importance of privacy policy from user's perspective. Last but not the

least, the more detailed functionalities, such as the share statement understanding, provide richer information digest from multiple dimensions. This helps to improve the usability of the system and meet different needs of the broad user group.

All in all, our approach in privacy policy evaluation does not rely on the adoptions by the websites. It opens the door for pure user-driven systems, which have huge potential to leverage the user's influence in order to change the landscape of the online privacy policy eco-system.

5.2 Limitations and future work

Current research is only the groundwork for this new area of study, despite the systems that are built and experimental results that are achieved. There are several limitations of the current work. Firstly, there are many important techniques in text classification that we have not applied yet. For example the feature projection techniques can be applied to further improve the systems. For both the task of privacy policy paragraph classification and the task of share statement understanding, the size of the corresponding datasets is limited due to the constraints of human resources. Richer datasets can help to increase the credibility of the experimental results. From software engineering point of view, though we build the systems by modules and intent to make them as easily re-usable as possible, the current systems are not developed by professional programmers and should be considered as prototypes. Also, because of the same constraints, less efforts are spent on elaborating the visualization and user interface. However, in practice, the good user interface and well-designed visualization scheme are crucial for the aforementioned concept of user-driven privacy policy evaluation.

We highlight several improvement areas for future work:

Current design of the systems focuses on the one-way interaction between the systems and the users. The main flow of information goes from the systems to the users, by presenting the automated evaluation results, including privacy policy topic viewing and gradings, to the users. An important further step is to build the functionality that facilitates the two-way interactions. Collaborative filtering is ideal for this purpose. Firstly, it is aligned with our generic machine learning based approach. Secondly, collaborative filtering is suitable for the grading functionality we introduced. A collaborative grading scheme of privacy policy can become the 'killer app' of our overall framework, that both largely leverage the user's influence

and greatly promote the awareness of privacy. Thirdly, collaborative filtering helps to enlarge the size of datasets and therefore optimize the models.

Many interesting relevant machine learning techniques have not been integrated into the current systems due to the limitation of resources. Feature projection techniques help to form a better feature space for text classification. Further research on implementing feature projection in current system can help to improve the text classification, especially for the privacy policy paragraph categorization task where the size of features is big. Techniques such as active learning and semi-supervised text classification can help to solve the problem of the limited datasets by either constantly receiving the newly labeled data from the users or integrating unlabeled data into the learning process. Another way to combine semantic information into the text classification is to use special kernels for SVM, such as string kernels, latent semantic kernel and syntactic tree kernel.

The share statement understanding chapter shows the potential to apply the generic machine learning approach to solve specific tasks in privacy policy evaluation. There are many possible patterns in privacy policies to be defined and solved using the similar approach. A future work is to systematically define these similar patterns, formally formulate the problems and apply the proposed methods.

In terms of software engineering, a systematic and professional implementation of the framework based on the current prototypes is the basis for future development of new functionalities and formal software performance research. Moreover, during this implementation, much more emphasis should be placed on the design and implementation of user interface and visualization methods. Especially, the design of grading visualization method is very important, as we consider the grading functionality to be the potential 'killer app' of the framework.

References

[1] California online privacy protection act. (2003). california business and professions code, sections 22575-22579. http://www.leginfo.ca.gov/cgi-bin/displaycode?section=bpc&group=22001-23000&file=22575-22579, 2003.

[2] P. Achananuparp, X. Hu, and X. Shen. The evaluation of sentence similarity measures. *Data Warehousing and Knowledge Discovery*, pages 305–316, 2008.

[3] P. Achananuparp, X. Hu, X. Zhou, and X. Zhang. Utilizing sentence similarity and question type similarity to response to similar questions in knowledge-sharing community. In *17th international conference on World Wide Web*, 2008.

[4] R. Agrawal, W.I. Grosky, and F. Fotouhi. Ranking privacy policy. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 192–197. IEEE, 2007.

[5] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 805–810. Lawrence Erlbaum Associates Ltd., 2003.

[6] R.M. Bell, Y. Koren, and C. Volinsky. The bellkor solution to the netflix prize. *KorBell Teams Report to Netflix*, 2007.

[7] C. M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[8] R.R. Bouckaert. Choosing between two learning algorithms based on calibrated tests. In *The Twentieth International Conference on Machine Learning (ICML-2003)*, volume 20, page 51, 2003.

[9] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[10] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[11] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112–116. Association for Computational Linguistics, 1992.

[12] G. Brown. Ensemble learning. *Encyclopedia of Machine Learnin*, 2009.

[13] A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*, volume 2, 2001.

[14] P. Buitelaar and P. Cimiano. *Ontology learning and population: bridging the gap between text and knowledge*, volume 167. Ios Pr Inc, 2008.

[15] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[16] D. Cai, S. Yu, J.R. Wen, and W.Y. Ma. Vips: a visionbased page segmentation algorithm. Technical report, Microsoft Technical Report, MSR-TR-2003-79, 2003.

[17] William B. Cavnar and John M. Trenkle. N-Gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US, 1994.

[18] Federal Trade Commission et al. Fair information practice principles. *Last Modified: Monday, June 25, 2007*.

[19] Federal Trade Commission et al. Childrens online privacy protection act of 1998. http://www.ftc.gov/ogc/coppa1.htm, 1998.

[20] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[21] E. Costante, J. den Hartog, and M. Petković. On-line trust perception: What really matters. In *Socio-Technical Aspects in Security and Trust (STAST), 2011 1st Workshop on*, pages 52–59. IEEE, 2011.

[22] T.N. Dao and T. Simpson. Measuring similarity between sentences. *online), http://wordnetdotnet.googlecode.com/svn/trunk/Projects/Thanh/Paper/WordNetDotNet_Semantic_Similarity.pdf,* {*Accessed on 2011-11-15*}, 2010.

[23] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM, 1998.

[24] H. Farrell. Constructing the international foundations of e-commercethe eu-us safe harbor arrangement. *International Organization*, 57(02):277–306, 2003.

[25] R. Feldman and J. Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge University Press, 2007.

[26] C. Fellbaum. Wordnet. *Theory and Applications of Ontology: Computer Applications*, pages 231–243, 2010.

[27] A. Finn, N. Kushmerick, and B. Smyth. Genre classification and domain transfer for information filtering. *Advances in Information Retrieval*, pages 349–352, 2002.

[28] P.W. Foltz, D. Laham, and T.K. Landauer. The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 1(2), 1999.

[29] The Center for Information Policy Leadership at Hunton & Williams LLP. Ten steps to develop a multilayered privacy notice. 2007.

[30] The Center for Information Policy Leadership at Hunton & Williams LLP. Multi-layered notices explained. *First Data Privacy Subgroup Meeting*, Feb. 2005.

[31] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.

[32] G. Forman. Bns feature scaling: an improved representation over tf-idf for svm text classification. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 263–270. ACM, 2008.

[33] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[34] A. Fujino, H. Isozaki, and J. Suzuki. Multi-label text categorization with model combination based on f1-score maximization. In *Proceedings of the 3rd International Joint Conference on Natural Language Porcessing*, pages 823–828, 2008.

[35] J. Furnkranz. A study using n-gram features for text categorization, 1998.

[36] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

[37] M.A. Graber, D.M. D Alessandro, and J. Johnson-West. Reading level of privacy policies on internet health web sites. *Journal of Family Practice*, 51(7):642–642, 2002.

[38] D.W. Harvey and A. White. Impact of computer security regulation on american companies, the. *Tex. Wesleyan L. Rev.*, 8:505, 2001.

[39] Trevor. Hastie, Robert. Tibshirani, and JH (Jerome H.) Friedman. *The elements of statistical learning*. Springer, 2009.

[40] M. Healy, S.J. Delany, and A. Zamolotskikh. An assessment of case base reasoning for short text message classification. In *Conference papers*, page 42, 2004.

[41] J.S. Hiller. The regulatory framework for privacy and security. *International Handbook of Internet Research*, pages 251–265, 2010.

[42] A.E. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, pages 55–67, 1970.

[43] C.W. Hsu, C.C. Chang, and C.J. Lin. A practical guide to support vector classification, 2010.

[44] C. Jensen and C. Potts. Privacy policies as decision-making tools: an evaluation of online privacy notices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 471–478. ACM, 2004.

[45] C. Jensen, C. Potts, and C. Jensen. Privacy practices of internet users: self-reports versus observed behavior. *International Journal of Human-Computer Studies*, 63(1):203–227, 2005.

[46] J. J. Jiang and D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, 1997.

[47] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Machine learning: proceedings of the fourteenth International Conference (ICML'97), Nashville, Tennessee, July 8-12, 1997*, page 143. Morgan Kaufmann Pub, 1997.

[48] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142, 1998.

[49] P.G. Kelley, J. Bresee, L.F. Cranor, and R.W. Reeder. A nutrition label for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, page 4. ACM, 2009.

[50] P.G. Kelley, L. Cesca, J. Bresee, and L.F. Cranor. Standardizing privacy notices: An online study of the nutrition label approach. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1573–1582. ACM, 2010.

[51] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145. Lawrence Erlbaum Associates Ltd., 1995.

[52] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450. ACM, 2010.

[53] C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[54] Y. Li, D. McLean, Z.A. Bandar, J.D. O'Shea, and K. Crockett. Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138–1150, 2006.

[55] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th international conference on machine learning*, volume 1, pages 296–304. San Francisco, 1998.

[56] R. Malik, L.V. Subramaniam, and S. Kaushik. Automatically selecting answer templates to respond to customer emails. In *IJCAI07–Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1659–1664, 2007.

[57] L.M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[58] C.D. Manning, P. Raghavan, and H. Schütze. Introduction to information retrieval. 2008.

[59] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[60] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

[61] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279. Association for Computational Linguistics, 2004.

[62] A. McDonald, R. Reeder, P. Kelley, and L. Cranor. A comparative study of online privacy policies and formats. In *Privacy Enhancing Technologies*, pages 37–55. Springer, 2009.

[63] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive bayes-which naive bayes. In *Third conference on email and anti-spam (CEAS)*, volume 17, pages 28–69, 2006.

[64] D. Metzler, Y. Bernstein, W.B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 517–524. ACM, 2005.

[65] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the national conference on artificial intelligence*, volume 21, page 775. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[66] G.A. Miller. Five papers on wordnet. *Technical Report CLS-Rep-43, Cognitive Science Laboratory, Princeton University*, 1993.

[67] Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 567–575, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[68] R. Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[69] A.Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[70] E.B. Page. The imminence ... of grading essays by computer. *The Phi Delta Kappan*, 47(5):238–243, 1966.

[71] E.B. Page and N.S. Petersen. The computer moves into essay grading: Updating the ancient test. *Phi Delta Kappan*, 76:561–561, 1995.

[72] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 2010, 2010.

[73] European Parliament and Council. Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Communities*, 23:31, 1995.

[74] European Parliament and Council. Directive 2006/24/ec of the european parliament and of the council of 15 march 2006 on the retention of data

generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending directive 2002/58/ec. *Official Journal of the European Communities*, pages 54–63, 2006.

[75] J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of the 18th international conference on World wide web*, pages 971–980. ACM, 2009.

[76] Jacob Perkins. Part of speech tagging with nltk part 4 - brill tagger vs classifier taggers. http://streamhacker.com/2010/04/12/pos-tag-nltk-brill-classifier, April 2010.

[77] M. Piotte and M. Chabbert. The pragmatic theory solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[78] S.P. Ponzetto and M. Strube. Knowledge derived from wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30(1):181–212, 2007.

[79] M.F. Porter et al. An algorithm for suffix stripping, 1980.

[80] OECD. Publishing. *OECD guidelines on the protection of privacy and transborder flows of personal data*. Organisation for Economic Co-operation and Development, 2002.

[81] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[82] J.R. Quinlan. *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[83] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. *Encyclopedia of Database Systems*, pages 532–538, 2009.

[84] TRUSTe Research. Privacy and online behavioral advertising - 2011 consumer research results. http://truste.com/ad-privacy/TRUSTe-2011-Consumer-Behavioral-Advertising-Survey-Results, July 2010.

[85] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.

[86] Ryan Michael Rifkin. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, 2002.

[87] L.M. Rudner and T. Liang. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2), 2002.

[88] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[89] S.L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and knowledge discovery*, 1(3):317–328, 1997.

[90] K.M. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 307–314. Association for Computational Linguistics, 2003.

[91] R.R. Schriver. You cheated, you lied: The safe harbor agreement and its enforcement by the federal trade commission. *Fordham L. Rev.*, 70:2777, 2001.

[92] S. Scott and S. Matwin. Text classification using wordnet hypernyms. In *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38–44, 1998.

[93] S. Scott and S. Matwin. Feature engineering for text classification. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, 1999.

[94] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[95] F. Sebastiani. Text categorization. *Text mining and its applications to intelligence, CRM and knowledge management*, pages 109–129, 2005.

[96] Stuart C. Shapiro. *Encyclopedia of artificial intelligence, Second Edition*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[97] G. Sigletos, G. Paliouras, C.D. Spyropoulos, and M. Hatzopoulos. Combining information extraction systems using voting and stacked generalization. *The Journal of Machine Learning Research*, 6:1751–1782, 2005.

[98] G.F. Smits and E.M. Jordaan. Improved svm regression using mixtures of kernels. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2785–2790. IEEE, 2002.

[99] P. Soucy and G.W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1130. Lawrence Erlbaum Associates Ltd., 2005.

[100] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[101] Hsuan tien Lin and Chih-Jen Lin. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. Technical report, National Taiwan University, March 2003.

[102] K.M. Ting and I.H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[103] M. Toman, R. Tesar, and K. Jezek. Influence of word normalization on text classification. *Proceedings of InSciT*, pages 354–358, 2006.

[104] J. Tsai, S. Egelman, L. Cranor, and A. Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. *Information Systems Research*, 21, 2010.

[105] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[106] T. Weninger and W.H. Hsu. Text extraction from the web via text-to-tag ratio. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, pages 23–28. IEEE, 2008.

[107] R. Wenning, M. Schunter, L. Cranor, M. Marchiori, et al. The platform for privacy preferences 1.1 (p3p1. 1) specification. *W3C Working Group Note*, 2006.

[108] M. Wiegand, A. Balahur, B. Roth, D. Klakow, and A. Montoyo. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68. Association for Computational Linguistics, 2010.

[109] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics, 2005.

[110] D.H. Wolpert. Stacked generalization*. *Neural networks*, 5(2):241–259, 1992.

[111] W. Wresch. The imminence of grading essays by computer-25 years later. *Computers and composition*, 10:45–45, 1993.

[112] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[113] D. Yang and D.M.W. Powers. Verb similarity on the taxonomy of wordnet. *Proceedings of GWC-06*, pages 121–128, 2006.

[114] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[115] S. Zelikovitz and H. Hirsh. Improving short text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1183–1190, 2000.

[116] S. Zelikovitz and H. Hirsh. Improving short-text classification using unlabeled data for classification problems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1191–1198, 2000.

[117] S. Zelikovitz and F. Marquez. Transductive learning for short-text classification problems using latent semantic indexing. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(2):143–164, 2005.

[118] J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 190–197. ACM, 2003.

[119] T. Zhang and F.J. Oles. Text categorization based on regularized linear classification methods. *Information retrieval*, 4(1):5–31, 2001.

[120] F. Zhou, Z. Fan, Y. Bingru, and Y. Xingang. Research on short text classification algorithm based on statistics and rules. In *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, pages 3–7. IEEE, 2010.

Appendix A

Implementation Details

The source codes of this thesis project are available on Github. Please visit https://github.com/YuanhaoSun for the details of the source codes.

In general, we built the programs in modules so as to provide flexibility for modification and future development. Another merit of the module structure is that functionalities provided by different modules can be combined in different ways to provide high extensibility.

The programs are mainly written in Python, with several R programs for experiment results plotting and few Java programs. Our programs reply heavily on the Python machine learning library, scikit-learn, which provides implementation of all kinds of classic machine learning algorithms.

The programs cover all three main parts of the implementation work of the thesis project, namely the topic classification, share statement understanding, and utilities.

A.1 Topic classification

Classification programs are implemented to apply different classifiers and ensemble methods on the datasets to learn the classification models.

Modules belong to this part are general classification module with different classifiers, cross-validation module, parameter tunning module, model storage module, grading module, multi-label classification module, two-layered classification module, visualization module that translates the results into xml files, stacking by predication ensemble module, stacking by probability score ensemble module, voting ensemble module.

A.2 Share statement understanding

Modules can be sorted into the two general approaches – the similarity based approach and the machine learning based approach.

For the similarity based approach, each similarity measure is implemented as a single module, including 3 modules of sentence overlap similarity measure, 4 modules of semantic similarity measures, 1 module of syntactic similarity measure and 1 module of hybrid measure. Two modules each implement the classification methods of averaging and ML-aided.

For the machine learning based approach, along with the similar modules in topic classification, modules for all the feature engineering methods are implemented.

A.3 Utilities

There are three different types of utility programs – the system utilities, the experiment utilities and the thesis utilities.

System utilities are the programs and code snippets built to support the core system functionality such as topic classification, grading, and share statement understanding. The accessorial components of the framework are all implemented as utility modules.

Experiment utilities are programs and code snippets facilitate the experiment processes, including modules that handle the file I/O, file type conversion, data format conversion, dataset conversion, text parsing, etc.

Thesis utilities are mainly the R snippets used to process and plot the raw experiment results.

Appendix B

Datasets

We construct several datasets to facilitate our research, especially the experimental evaluations. Please visit https://github.com/YuanhaoSun/MLToolkit/tree/master/Dataset for the datasets.

Table B.1: General information of datasets

Name	Section	Size	Description
Paragraphs training set	3.3.1	772	Training set for topic classification, with sub-category labels
Paragraphs test set	3.3.1	277	Test set for topic classification, with sub-category labels
Share statement training set	4.4.1	75	Training set for the task of share statement understanding
Share statement test set	4.4.1	37	Test set for the task of share statement understanding
Full-text privacy policy dataset	2.2.1	934	Full-text privacy policies search results, generated from detector test
Pure privacy policy dataset	2.2.1	796	All privacy policies, derived from 934 full-text set
Grading dataset	3.4.3.3	126	Grading of full-text privacy policies, manually grading with automatically generated scores

Coverage of share statement datasets The sizes of the share statement datasets are relatively small compared with the privacy policy paragraphs datasets. This is caused by the nature of the share statements. The share statements are short text containing very limited amounts of vocabularies and expressing very similar

meanings. The share statements in one specific categories can be highly resemblant to each other.

In a Google search statistics study, we test the first 30 items in the 75 share statement training set, and search the extract matches and partial sentence matches. There are many share statements that have exact matches or almost-exact matches up to tens of thousands on the Internet.

For example, the share statement "We may disclose your personal information to third parties" has around 2,750,000 exact matches in Google search results, "We do not sell your personal information to third parties." has 1,470,000, "We neither rent nor sell your Personal Information to anyone." has 38,400, "We never share or sell your personal information" has 49,700.

All in all, the limited size of share statement datasets are constrained by the nature of share statements.

Two examples In order to fill at least one test example in each category in the share statement test dataset (see table 4.5), we created two examples for the category 'Not sell Not share' and the category 'Sell and share'.

"We may share or disclose your personal information to third parties" is the sentence created as a test example in category 'Not sell Not share'.

"We reserves the rights to rent, sell, lease or otherwise share or disclose your personal information to any third parties" is the sentence created as a test example in category 'Sell and share'.

The reasons for creating these two examples are: 1) At the time of building the training set, we had not labeled all items in the 95 privacy policy set. This also complies to rule of thumb which requires not knowning the test case during training. After we almost finished the training experiment, we finished the labeling of all items. It turned out to be that there is no example for these two categories. 2) However, in order to reflect the categorical distribution of real life test data, we have already limited the scope of test to be the 95 privacy policy set.

Therefore, we created two examples for the two categories which were empty. These two sentences were drafted in a manner that largely keeps the consistency with other examples possible yet avoids pure duplication. A strong proof of their validation is that there are exact match of the first example on the Internet. Though there is no exact match for the second due to the scarcity of this categories, there are close matches which have similar meanings.

Appendix C

Experiment Details

C.1 Parameter tuning of classifiers

As elaborated in section 3.3.4.4, parameter tuning leads to further optimized performance of the classifiers that have alterable parameter(s). In this part of appendix, we provide the detailed information about the process of parameter tunning. To point out, we only tune the parameter on the full sized training set, that is, the training set with 772 as introduced in section 3.3.1 because only one set of tunned paramters can be applied to the test set. Further, grid search is normally carried out based on one metric. For this, we will select the precision as the metric to tune the classifier.

For **kNN**, we tune the parameter of k. One specific trick for a rough k estimation is to choose $k = \frac{\#\text{feature}}{2*\#\text{categories}}$ which we have used in the comparison where the precision scores are given by the change of size of training set. We will, however, test k using grid search based on the trainig set. The result is shown in the following table:

Table C.1: Parameter tuning - kNN

k	F_1	k	F_1	k	F_1	k	F_1
1	0.7403	3	0.7917	5	0.8127	7	0.8278
9	0.8347	11	0.8392	13	0.8481	**15**	**0.8506**
17	0.8487	**19**	**0.8501**	21	0.8452	23	0.8493
25	0.8431	27	0.8482	29	0.8399	31	0.8389
33	0.8278	35	0.8258	37	0.8171	39	0.8101
41	0.8065	43	0.8006	45	0.8002	47	0.7946
49	0.7995	51	0.7923	53	0.7902	55	0.7807

As shown in the table, the performance of kNN reaches the peak around $k = 15, 17, 19$. We will therefore choose $k = 19$ as it is also closer to the practical trick where $k = \frac{\#\text{feature}}{2*\#\text{categories}}$.

SVM has two parameters mainly for tuning, i.e. γ and C. One practical trick for tuning γ is to choose$\gamma = \frac{1}{\#\text{feature}}$. The practical guide in tunning the SVM provided in paper [43] is widely cited and applied. We follow this guide by first applying a loose search and then going into a fine search.

In the loose search, we set $C = \{2^{-3}, 2^{-1}, 2, 2^3, 2^5, 2^7, 2^9, 2^{11}, 2^{13}, 2^{15}\}$ and $\gamma = \{2^{-15}, 2^{-11}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}\}$. The grid search tests all the combinations between the available values of C and γ. The best performance is generated when $C = 2^7$ and $\gamma = 2^{-5}$. Further, we carry out a fine search near the best combination with $C = \{2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}\}$ and $\gamma = \{2^{-8}, 2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}\}$. Then the final selected parameters are $C = 2^5$ and $\gamma = 2^{-4}$.

Similarly, for **linear SVM**, we tune the parameter C for both $l1$ regularization and $l2$ regularization. The parameter space for $C = \{2^{-3}, 2^{-1}, 2, 2^3, 2^5, 2^7, 2^9, 2^{11}, 2^{13}, 2^{15}\}$. The grid search shows $C = 0.5$ when using $l2$ regularization yields the best performance.

For **decision tree**, we test the parameter space with max_depth = $\{6, 8, 10, 12, 14, 16\}$ and min_split = $\{2, 3, 4, 5, 6\}$. The grid search finds the best parameter combination with max_depth = 16 and min_split = 5.

C.2 List of share statement final test results

Table C.2: Comparison of two share statement classification approaches - comprehensive

Type	**Classification / Clssifer**	**Method**	**3-class task**		**8-class task**	
			F_1	$F_{0.5}$	F_1	$F_{0.5}$
Similarity based	Averaging	Overl.	87.05	89.14	58.76	67.14
		IDF	87.83	92.44	64.65	70.54
		Phra.	85.05	87.63	48.19	51.75
		Alpha	52.77	69.13	37.73	41.83
		POS	37.94	57.12	24.17	24.09
		Int.Max	58.30	69.52	46.11	53.07
		WrdOr.	70.40	79.14	65.95	70.58
		Hy_0.8	62.19	74.66	64.01	71.95
		Hy_0.5	72.47	80.48	66.33	71.27
	ML aided	Overl.	90.43	91.19	73.90	76.27
		IDF	93.22	94.04	60.81	63.53
		Phra.	82.02	82.59	52.20	54.59
		Alpha	81.49	83.48	52.89	55.81
		POS	89.81	91.93	79.76	80.61
		Int.Max	79.45	80.10	49.90	47.69
		WrdOr.	81.23	83.03	71.48	74.05
		Hy_0.8	83.93	85.74	78.90	80.97
		Hy_0.5	85.02	85.78	71.54	74.16
Machine Learning based (Max)	Naive Bayes	Baseline	89.19	89.19	79.45	78.51
		Neg. Bi.	86.85	87.06	72.58	74.03
		POS words	89.19	89.19	77.82	77.49
		Combined	91.52	91.31	74.84	74.35
	Ridge	Baseline	80.83	84.27	64.71	68.34
		Neg. Bi.	80.83	84.27	73.08	74.88
		POS words	80.83	84.27	78.46	80.23
		Combined	84.43	88.54	72.88	74.54
	SVM	Baseline	90.60	91.47	78.04	79.53
		Neg. Bi.	86.61	86.69	72.15	73.33
		POS words	91.52	91.31	81.26	81.37
		Combined	92.25	92.46	77.34	78.37
Machine Learning based (100%)	Naive Bayes	Baseline	87.20	87.64	71.81	72.77
		Neg. Bi.	86.85	87.06	68.46	69.00
		POS words	89.19	89.19	77.82	77.49
		Combined	89.19	89.19	74.84	74.35
	Ridge	Baseline	75.37	80.97	64.71	68.34
		Neg. Bi.	77.33	82.29	69.53	72.66
		POS words	77.33	82.29	74.36	77.06
		Combined	87.77	84.94	68.51	70.90
	SVM	Baseline	80.09	81.15	78.04	79.53
		Neg. Bi.	82.50	83.37	68.29	70.53
		POS words	87.20	87.64	81.26	81.37
		Combined	91.52	91.31	76.42	76.87

C.3 Common English Stopwords and POS tag set

['all', 'six', 'less', 'being', 'indeed', 'over', 'move', 'anyway', 'four', 'not', 'own', 'through', 'yourselves', 'fify', 'where', 'mill', 'only', 'find', 'before', 'one', 'whose', 'system', 'how', 'somewhere', 'with', 'thick', 'show', 'had', 'enough', 'should', 'to', 'must', 'whom', 'seeming', 'under', 'ours', 'has', 'might', 'thereafter', 'latterly', 'do', 'them', 'his', 'around', 'than', 'get', 'very', 'de', 'none', 'cannot', 'every', 'whether', 'they', 'front', 'during', 'thus', 'now', 'him', 'nor', 'name', 'several', 'hereafter', 'always', 'who', 'cry', 'whither', 'this', 'someone', 'either', 'each', 'become', 'thereupon', 'sometime', 'side', 'two', 'therein', 'twelve', 'because', 'often', 'ten', 'our', 'eg', 'some', 'back', 'up', 'go', 'namely', 'towards', 'are', 'further', 'beyond', 'ourselves', 'yet', 'out', 'even', 'will', 'what', 'still', 'for', 'bottom', 'mine', 'since', 'please', 'forty', 'per', 'its', 'everything', 'behind', 'un', 'above', 'between', 'it', 'neither', 'seemed', 'ever', 'across', 'she', 'somehow', 'be', 'we', 'full', 'never', 'sixty', 'however', 'here', 'otherwise', 'were', 'whereupon', 'nowhere', 'although', 'found', 'alone', 're', 'along', 'fifteen', 'by', 'both', 'about', 'last', 'would', 'anything', 'via', 'many', 'could', 'thence', 'put', 'against', 'keep', 'etc', 'amount', 'became', 'ltd', 'hence', 'onto', 'or', 'con', 'among', 'already', 'co', 'afterwards', 'formerly', 'within', 'seems', 'into', 'others', 'while', 'whatever', 'except', 'down', 'hers', 'everyone', 'done', 'least', 'another', 'whoever', 'moreover', 'couldnt', 'throughout', 'anyhow', 'yourself', 'three', 'from', 'her', 'few', 'together', 'top', 'there', 'due', 'been', 'next', 'anyone', 'eleven', 'much', 'call', 'therefore', 'interest', 'then', 'thru', 'themselves', 'hundred', 'was', 'sincere', 'empty', 'more', 'himself', 'elsewhere', 'mostly', 'on', 'fire', 'am', 'becoming', 'hereby', 'amongst', 'else', 'part', 'everywhere', 'too', 'herself', 'former', 'those', 'he', 'me', 'myself', 'made', 'twenty', 'these', 'bill', 'cant', 'us', 'until', 'besides', 'nevertheless', 'below', 'anywhere', 'nine', 'can', 'of', 'toward', 'my', 'something', 'and', 'whereafter', 'whenever', 'give', 'almost', 'wherever', 'is', 'describe', 'beforehand', 'herein', 'an', 'as', 'itself', 'at', 'have', 'in', 'seem', 'whence', 'ie', 'any', 'fill', 'again', 'hasnt', 'inc', 'thereby', 'thin', 'no', 'perhaps', 'latter', 'meanwhile', 'when', 'detail', 'same', 'wherein', 'beside', 'also', 'that', 'other', 'take', 'which', 'becomes', 'you', 'if', 'nobody', 'see', 'though', 'may', 'after', 'upon', 'most', 'hereupon', 'eight', 'but', 'serious', 'nothing', 'such', 'your', 'why', 'a', 'off', 'whereby', 'third', 'i', 'whole', 'noone', 'sometimes', 'well', 'amoungst', 'yours', 'their', 'rather', 'without', 'so', 'five', 'the', 'first', 'whereas', 'once']

Table C.3: Penn Treebank II POS tag set

Tag	Description	Example
CC	conjunction, coordinating	and, or, but
CD	cardinal number	five, three, 13%
DT	determiner	the, a, these
EX	existential there	there were six boys
FW	foreign word	mais
IN	conjunction, subordinating or preposition	of, on, before, unless
JJ	adjective	nice, easy
JJR	adjective, comparative	nicer, easier
JJS	adjective, superlative	nicest, easiest
LS	list item marker	
MD	verb, modal auxillary	may, should
NN	noun, singular or mass	tiger, chair, laughter
NNS	noun, plural	tigers, chairs, insects
NNP	noun, proper singular	Germany, God, Alice
NNPS	noun, proper plural	we met two Christmases ago
PDT	predeterminer	both his children
PRP	pronoun, personal	me, you, it
PRP$	pronoun, possessive	my, your, our
RB	adverb	extremely, loudly, hard
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	adverb, particle	about, off, up
SYM	symbol	%
TO	infinitival to	what to do?
UH	interjection	oh, oops, gosh
VB	verb, base form	think
VBZ	verb, 3rd person singular present	she thinks
VBP	verb, non-3rd person singular present	I think
VBD	verb, past tense	they thought
VBN	verb, past participle	a sunken ship
VBG	verb, gerund or present participle	thinking is fun
WDT	wh-determiner	which, whatever, whichever
WP	wh-pronoun, personal	what, who, whom
WP$	wh-pronoun, possessive	whose, whosever
WRB	wh-adverb	where, when
.	punctuation mark, sentence closer	.;?*
,	punctuation mark, comma	,
:	punctuation mark, colon	:
(	contextual separator, left paren	(
)	contextual separator, right paren	)

Accessed from http://www.cnts.ua.ac.be/pages/mbsp-tags