Quality-Efficiency Trade-offs in Machine Learning Applied to Text Processing

Zeinab Liaghat

TESI DOCTORAL UPF / 2016-2017

DIRECTOR DE LA TESI
Dr. Ricardo Baeza-Yates

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGIES

Acknowledgements

I would like to take this opportunity to say thank you to my advisor, Prof. Ricardo Baeza-Yates for the continuous provision of my Ph.D. research, for his useful comments, motivation and engagement through the different stages of this thesis. Without his valuable assistance, I could not have completed this research. He has been tireless and kind in helping to deal with difficult problems over the course of these three years.

I would like to thank my family for their unending support throughout my research both spiritually and in general. My father has been my main motivation to achieve my PhD. My husband has supported me during my entire studies and has been most understanding and has endured the whole process with little or no complaints. I owe him a great deal for this.

Summary

Nowadays, the amount of available digital documents is rapidly growing, expanding at a considerable rate and coming from a variety of sources. Sources of unstructured and semi-structured information include the World Wide Web, news articles, biological databases, electronic mail, digital libraries, governmental digital repositories, chat rooms, online forums, blogs, and social media such as Facebook, Instagram, LinkedIn, Pinterest, Twitter, YouTube, Instagram, Pinterest, plus many others.

Extracting information from these resources and finding useful information from such collections has become a challenge, which makes organizing massive amounts of data a necessity. Data mining, machine learning, and natural language processing are powerful techniques that can be used together to deal with this big challenge. Depending on the task or problem at hand, there are many different approaches that can be used. The methods that are being implemented are continuously being optimized, but not all these methods have been tested and compared for quality after training on large size corpora for supervised machine learning algorithms. The question is what happens to the quality of methods if we increase the data size from, say, 100 MB to over 1 GB? Moreover, are quality gains worth it when the rate of data processing diminishes? Can we trade quality for time efficiency and recover the quality loss by just being able to process more data?

This thesis is first attempt to answer these questions in a general way for text processing tasks, as not enough research has been done to compare those methods considering the trade-offs of data size, quality, and processing time. Hence, we propose a trade-off analysis framework and apply it to three important text processing problems: Named Entity Recognition, Sentiment Analysis, and Document Classification. These problems were also chosen because they have different levels of object granularity: words, passages, and documents. For each problem, we select several machine learning algorithms and we evaluate the trade-offs of these different methods on large publicly available datasets (news, reviews, patents). We use different data subsets of increasing size ranging from 50 MB to a few GB, to explore these trade-offs. We conclude, as hypothesized, that just because the method has good performance in small data, it does not necessarily have the same performance for big data. For the two last problems, we consider similar algorithms and also consider two different data sets and two different evaluation techniques, to study the impact of the data and the evaluation technique on the resulting trade-offs. We find that the results do not change significantly.

Resum

Avui en dia, la quantitat de documents digitals disponibles està creixent ràpidament, expandint-se a un ritme considerable i procedint de diverses fonts. Les fonts d'informació no estructurada i semiestructurada inclouen la World Wide Web, articles de notícies, bases de dades biològiques, correus electrònics, biblioteques digitals, repositoris electrònics governamentals, , sales de xat, forums en línia, blogs i mitjans socials com Facebook, Instagram, LinkedIn, Pinterest, Twitter, YouTube i molts d'altres.

Extreure'n informació d'aquests recursos i trobar informació útil d'aquestes col.leccions s'ha convertit en un desafiament que fa que l'organització d'aquesta enorme quantitat de dades esdevingui una necessitat. La mineria de dades, l'aprenentatge automàtic i el processament del llenguatge natural són tècniques poderoses que poden utilitzar-se conjuntament per fer front a aquest gran desafiament. Segons la tasca o el problema en qüestió existeixen molts emfocaments diferents que es poden utilitzar. Els mètodes que s'estan implementant s'optimitzen continuament, però aquests mètodes d'aprenentatge automàtic supervisats han estat provats i comparats amb grans dades d'entrenament. La pregunta és : Què passa amb la qualitat dels mètodes si incrementem les dades de 100 MB a 1 GB? Més encara: Les millores en la qualitat valen la pena quan la taxa de processament de les dades minva? Podem canviar qualitat per eficiència, tot recuperant la perdua de qualitat quan processem més dades?

Aquesta tesi és una primera aproximació per resoldre aquestes preguntes de forma general per a tasques de processament de text, ja que no hi ha hagut suficient investigació per a comparar aquests mètodes considerant el balanç entre el tamany de les dades, la qualitat dels resultats i el temps de processament. Per tant, proposem un marc per analitzar aquest balanç i l'apliquem a tres problemes importants de processament de text: Reconeixement d'Entitats Anomenades, Anàlisi de Sentiments i Classificació de Documents. Aquests problemes també han estat seleccionats perquè tenen nivells diferents de granularitat: paraules, opinions i documents complerts. Per a cada problema seleccionem diferents algoritmes d'aprenentatge automàtic i avaluem el balanç entre aquestes variables per als diferents algoritmes en grans conjunts de dades públiques (notícies, opinions, patents). Utilitzem subconjunts de diferents tamanys entre 50 MB i alguns GB per a explorar aquests balanç. Per acabar, com havíem suposat, no perquè un algoritme és eficient en poques dades serà eficient en grans quantitats de dades. Per als dos últims problemes considerem algoritmes similars i també dos conjunts diferents de dades i tècniques d'avaluació per a estudiar l'impacte d'aquests dos paràmetres en els resultats. Mostrem que els resultats no canvien significativament amb aquests canvis.

Resumen

Hoy en día, la cantidad de documentos digitales disponibles está creciendo rápidamente, expandiéndose a un ritmo considerable y procediendo de una variedad de fuentes. Estas fuentes de información no estructurada y semi estructurada incluyen la World Wide Web, artículos de noticias, bases de datos biológicos, correos electrónicos, bibliotecas digitales, repositorios electrónicos gubernamentales, salas de chat, foros en línea, blogs y medios sociales como Facebook, Instagram, LinkedIn, Pinterest, Twitter, YouTube, además de muchos otros.

Extraer información de estos recursos y encontrar información útil de tales colecciones se ha convertido en un desafío que hace que la organización de esa enorme cantidad de datos sea una necesidad. La minería de datos, el aprendizaje automático y el procesamiento del lenguaje natural son técnicas poderosas que pueden utilizarse conjuntamente para hacer frente a este gran desafío. Dependiendo de la tarea o el problema en cuestión, hay muchos enfoques diferentes que se pueden utilizar. Los métodos que se están implementando se están optimizando continuamente, pero estos métodos de aprendizaje automático supervisados han sido probados y comparados con datos de entrenamiento grandes. La pregunta es ¿Qué pasa con la calidad de los métodos si incrementamos los datos de 100 MB a 1GB? Más aún, ¿las mejoras en la calidad valen la pena cuando la tasa de procesamiento de los datos disminuye? ¿Podemos cambiar calidad por eficiencia, recuperando la perdida de calidad cuando procesamos más datos?

Esta tesis es una primera aproximación para resolver estas preguntas de forma general para tareas de procesamiento de texto, ya que no ha habido investigación suficiente para comparar estos métodos considerando el balance entre el tamaño de los datos, la calidad de los resultados y el tiempo de procesamiento. Por lo tanto, proponemos un marco para analizar este balance y lo aplicamos a tres importantes problemas de procesamiento de texto: Reconocimiento de Entidades Nombradas, Análisis de Sentimientos y Clasificación de Documentos. Estos problemas fueron seleccionados también porque tienen distintos niveles de granularidad: palabras, opiniones y documentos completos. Para cada problema seleccionamos distintos algoritmos de aprendizaje automático y evaluamos el balance entre estas variables para los distintos algoritmos en grandes conjuntos de datos públicos (noticias, opiniones, patentes). Usamos subconjuntos de distinto tamaño entre 50 MB y varios GB para explorar este balance. Para concluir, como habíamos supuesto, no porque un algoritmo es eficiente en pocos datos será eficiente en grandes cantidades de datos. Para los dos últimos problemas consideramos algoritmos similares y también dos conjuntos distintos de datos y técnicas de evaluación, para estudiar el impacto de estos dos parámetros en los resultados. Mostramos que los resultados no cambian significativamente con estos cambios.

خلاصه

امروزه، افزایش میزان در دسترس بودن اسناد الکترونیکی که از منابع مختلف جمع می شود، به سرعت و با یک نرخ قابل توجه در حال رشد و گسترش است. نوع این اطلاعات می تواند نیمه ساختاری یا بدون ساختار باشد که شامل شبکه جهانی وب، مقالات، اخبار، پایگاه داده های بیولوژیکی، پست الکترونیکی، کتابخانه دیجیتال، مخازن دیجیتالی دولتی، اتاق های گفتگو، انجمن های آنلاین، وبلاگ ها، و رسانه های اجتماعی مانند فیس بوک، نمایش مشخصات عمومی، لینک، پینترست ، توییتر ، یوتیوب، نمایش مشخصات عمومی، پینترست، و خیلی های دیگر.

استخراج اطلاعات از این منابع و پیدا کردن اطلاعات مفید از ان مجموعه یک چالش بزرگ است، که باعث می شود سازماندهی حجم انبوهی از داده، یک ضرورت شناخته شود. داده کاوی، یادگیری ماشین و پردازش زبان طبیعی تکنیک های قدرتمند است که می تواند با هم برای مقابله با این چالش بزرگ مورد استفاده واقع شود. بستگی به موضوع و یا مسئله، روش های مختلف داده کاوی، یادگیری ماشین, یا پردازش زبان طبیعی به طور مداوم در حال بهینه سازی می باشد. اما کیفیت همه این روش ها تست و مقایسه نشده در زمانی که الگوریتم روی داده با هجم زیاد آموزش داده شده باشد.
سوال این است که چه اتفاقی برای کیفیت روش ها می افتد اگر ما حجم داده رو افزایش بدیم، برای مثال از ۱۰۰MB به ۱GB؟ علاوه بر این، آیا ارزش زمانی نرخ پردازش داده ،دستاوردهای کیفیت را کاهش می دهد؟ آیا ما می توانیم کیفیت را با بهره وری زمان مبادله کنیم و بازیابی از دست دادن کیفیت را فقط توسط قدرت پردازش داده های بیشتر برگردانیم.

این پایان نامه ما تلاش میکنیم برای پیدا کردن جواب عمومی به این پرسش درپردازش متن. تحقیقات زیادی روی مقایسه روش ها مختلف با درنظر اندازه داده، کیفیت و زمان پردازش صورت نگرفته است. از این رو، ما یک چارچوب پیشنهاد دادیم که بتوانیم تجزیه و تحلیل کنیم رفتار های الگوریتم ها را روی سه تا مساله مهم پردازش متن: استخراج کلمه، تجزیه و تحلیل احساسات و طبقه بندی سند. ما انتخاب کردیم این سه مساله برای اینکه اینها دارای سطوح مختلف هستند: کلمات، عبارات و اسناد. برای هر مشکل ما چندین الگوریتم یادگیری ماشین را انتخاب کردیم و ما ارزیابی کردیم نتایج این روش های مختلف را بر روی مجموعه داده های بزرگ در دسترس عموم (اخبار، بررسی، اختراع ثبت شده). با استفاده از زیر مجموعه های داده مختلف با حجم های متفاوت، از ۵۰ مگابیت تا چند گیگابایت، رفتار الگوریتم ها را مقایسه کردیم. ما نتیجه گرفتیم ، به عنوان فرضیه، که فقط به خاطر اینکه یک روش دارای عملکرد خوب در داده کوچک است، لزوما عملکرد مشابه بر روی داده های بزرگ را نشان نمی دهد. برای دو مساله آخر، ما الگوریتم های مشابه و دو مجموعه داده های مختلف را در نظر گرفته ایم، که بتوانیم روی تاثیر مساله و تاثیر داده مطالعه کنیم. در پایان ما متوجه شدیم که نتایج نهایی به طور قابل توجهی را تغییر نکرده است.

Contents

List of Figures **xii**

List of Tables **xiv**

1 INTRODUCTION **1**
1.1 Motivation . . . 1
1.2 Formal Problem . . . 3
1.3 Contributions . . . 4
1.4 Organization . . . 5

2 BACKGROUND **7**
2.1 Introduction . . . 7
2.2 Supervised Machine Learning Algorithms . . . 8
2.2.1 Logistic Regression . . . 8
2.2.2 Decision Trees . . . 8
2.2.3 K-Nearest Neighbors . . . 9
2.2.4 Support Vector Machines . . . 10
2.2.5 Naïve Bayes Classifier . . . 11
2.2.6 Random Forest . . . 11
2.2.7 Conditional Random Fields . . . 13
2.2.8 Multilayered Neural Networks . . . 13
2.2.9 Hidden Markov Models . . . 14
2.2.10 Language Models . . . 14
2.3 Classical Document Features . . . 14
2.3.1 Term Frequency . . . 14
2.3.2 Inverse Document Frequency . . . 15
2.3.3 Term Frequency-Inverse Document Frequency . . . 17
2.4 Performance Trade-offs in Machine Learning . . . 18
2.5 Named Entity Recognition . . . 24
2.6 Sentiment Analysis . . . 27
2.7 Document Classification . . . 29

3 TRADE-OFFS FRAMEWORK **35**
3.1 Introduction . . . 35
3.2 Measuring Quality . . . 35
3.2.1 Confusion Matrix . . . 35
3.2.2 Precision and Recall . . . 36
3.2.3 F-measure . . . 37
3.3 Time Efficiency . . . 38

3.4 Trade-off Analysis . . . 39
3.4.1 Quality and Data Size . . . 41
3.4.2 Time and Data size . . . 41
3.4.3 Quality, Time and Data Size . . . 41
3.4.4 Dominant Algorithms . . . 42
3.5 Methodology . . . 44
3.5.1 Data Collection . . . 44
3.5.2 Defining the Target Variable . . . 44
3.5.3 Pre-process Data . . . 45
3.5.4 Classification Methods . . . 45
3.5.5 Evaluation Metrics and Validation . . . 45
3.5.6 Trade-off Analysis . . . 46
3.6 Discussion . . . 47

4 NAMED ENTITY RECOGNITION 49
4.1 Introduction . . . 49
4.2 Algorithms . . . 50
4.2.1 Stanford NER . . . 51
4.2.2 Illinois Named Entity Tagger . . . 52
4.2.3 LingPipe . . . 52
4.3 Dataset: News . . . 52
4.4 Experimental Setup . . . 55
4.4.1 Extractor . . . 55
4.4.2 Searcher . . . 55
4.4.3 Disambiguator . . . 55
4.5 Experimental Results . . . 57
4.5.1 Quality Comparison . . . 57
4.5.2 Time Efficiency Comparison . . . 58
4.5.3 Dominant Algorithm . . . 58
4.5.4 Overall Comparison . . . 59
4.6 Discussion . . . 60

5 SENTIMENT ANALYSIS 61
5.1 Introduction . . . 61
5.2 Dataset: Amazon Reviews . . . 61
5.3 Movies and Television . . . 62
5.3.1 Dataset: Movies & TV Reviews . . . 62
5.3.2 Experimental Setup . . . 64
5.3.3 Experimental Results . . . 66
5.4 Books . . . 71
5.4.1 Dataset: Books Reviews . . . 71
5.4.2 Experimental Setup . . . 72
5.4.3 Experimental Results . . . 75
5.5 Discussion . . . 76

6 DOCUMENT CLASSIFICATION **79**
6.1 Introduction . . . 79
6.2 Algorithms . . . 80
6.3 News Classification . . . 81
6.3.1 Dataset: News . . . 81
6.3.2 Experimental Setup . . . 81
6.3.3 Experimental Results . . . 86
6.3.4 Analysis . . . 90
6.4 Patents Classification . . . 91
6.4.1 Dataset: USA Patents . . . 91
6.4.2 Experimental Setup . . . 93
6.4.3 Experimental Results . . . 95
6.4.4 Analysis . . . 98
6.5 Discussion . . . 99

7 CONCLUSIONS AND FUTURE WORK **101**
7.1 Conclusions . . . 101
7.2 Future Work . . . 102

Bibliography **105**

List of Figures

1.1 Machine learning workflow process. 2
1.2 Knowledge discovery process. 3

2.1 Complexity of language technology problems [Jurafsky and Martin, 2000]. . . 7
2.2 Logistic regression example. 8
2.3 Decision trees example. 9
2.4 K-nearest neighbors example [Wikipedia, 2017a]. 9
2.5 Support vector machine example [Wikipedia, 2017b]. 10
2.6 Examples of multilayer neural network architectures [Schmidt and Okt, 2000]. 13
2.7 Accuracy vs. speed for SD parsing. 21
2.8 Voting among classifiers [Banko and Brill, 2001]. 22
2.9 Active learning with large corpora [Banko and Brill, 2001]. 23
2.10 Learning Curves for Confusable Disambiguation [Banko and Brill, 2001]. . . . 23
2.11 An example of NER system. 25
2.12 Impact of data size [Ji and Grishman, 2006]. 27
2.13 Polarity of all methods across the labelled datasets [Goncalves et al., 2013]. . . 28
2.14 Coverage and F-measure comparison for all methods [Gonçalves et al., 2013]. . 29
2.15 Coverage vs F-measure trade-off for all methods [Gonçalves et al., 2013]. . . . 29
2.16 Data based on product categories [Fang and Zhan, 2015]. 30
2.17 Data based on review categories [Fang and Zhan, 2015]. 30
2.18 F1 Score of review-level categorization [Fang and Zhan, 2015]. 31
2.19 The text classification process. 32

3.1 Running time vs. data size. 41
3.2 Accuracy on basic dependencies. 42
3.3 Accuracy vs. tokens per second. 43
3.4 Performance vs. data size. 43
3.5 Example for dominant algorithms. 44
3.6 Example of a 5-fold cross validation scheme. 46
3.7 Knowledge discovery process. 47

4.1 Problems that need NER. 49
4.2 Example of the CoNLL03 dataset. 54
4.3 Sample of tokenized output generated by a system. 54
4.4 Sample clean tokenized. 56
4.5 Sample of output solution [Nadeau and Sekine, 2007]. 56
4.6 Sample of output from a system [Nadeau and Sekine, 2007]. 56
4.7 Comparison of recognizer based on different data sizes on CoNLL. 57
4.8 Running time vs. data size for all NER systems. 59
4.9 Dominant algorithm for NER systems on 100 MB. 59

4.10 Dominant algorithm for NER systems on 500 MB. 60
4.11 Performance and data size in NER. 60

5.1 Semantic analysis workflow. 62
5.2 Rating system for Amazon.com. 62
5.3 Score distribution before normalization. 65
5.4 Sample of dataset after normalization. 66
5.5 Comparison of algorithms quality on different data sizes. 67
5.6 Time efficiency vs. data size. 68
5.7 Dominant algorithms on Sentiment Analysis for 50 MB. 68
5.8 Dominant algorithms on Sentiment Analysis for 500 MB. 69
5.9 Dominant algorithms on Sentiment Analysis for 1.2 GB. 69
5.10 Comparing three algorithms in time and quality. 70
5.11 Performance comparison of the algorithms on different data sizes. 70
5.12 Score distribution of Amazon reviews. 72
5.13 Sample of data after target variables. 72
5.14 Score distribution before normalization. 74
5.15 Comparison for quality on different data sizes. 75
5.16 Comparison for time efficiency on different data sizes. 76
5.17 Dominant algorithms on Sentiment Analysis for 10 MB. 76
5.18 Dominant algorithms on Sentiment Analysis for 50 MB. 77
5.19 Dominant algorithms on Sentiment Analysis for 3 GB. 77
5.20 Performance comparison for different data sizes. 78

6.1 Sample of XML file. 82
6.2 Sample of data after parsing. 83
6.3 Document classification process. 84
6.4 Frequency of topics in RCV1. 84
6.5 Frequency of selected topics in our dataset. 85
6.6 Quality vs. Data size. 87
6.7 Time efficiency vs. Data size. 88
6.8 Dominant algorithms on news classification for 500 MB data size. 88
6.9 Dominant algorithms for news classification for 1.2 GB data size. 89
6.10 Dominant algorithms for news classification for 1.5 GB data size. 89
6.11 Comparing the performance of all algorithms for news classification. 90
6.12 Example of patent metadata. 92
6.13 Data sample example. 94
6.14 Quality vs. Data size. 95
6.15 Time Efficiency vs. Data size. 96
6.16 Dominant algorithms on patents classification for 100 MB. 96
6.17 Dominant algorithms on patents classification for 500 MB. 97
6.18 Dominant algorithms on patents classification for 1 GB. 97
6.19 Dominant algorithms on patents classification for 4 GB. 98
6.20 Dominant algorithms on patents classification for 7.18 GB. 98
6.21 Comparing the performance of all algorithms for patents classification. 99

List of Tables

2.1 Training dataset. 15
2.2 Test dataset. 15
2.3 Performance comparison of different classification algorithms [Gandhi and Prajapati, 2012]. 19
2.4 Unlabeled and labelled attachment F1 score and time "to generate standard Stanford dependencies with different types of parsers" [Cer et al., 2010]. 20
2.5 Basic SD parsing performance and running time [Kong and Smith, 2014]. . . . 20
2.6 F-measures for the eight methods [Gonçalves et al., 2013]. 28

3.1 Contingency table binary categorization quality for a category. 36
3.2 Document classification confusion matrix. 36
3.3 Table of confusion for the sport's class. 37
3.4 Sport class. 38
3.5 Economic class. 38
3.6 Micro Average Table. 38
3.7 Example running time of two algorithms A and B. 39
3.8 Example running time for algorithms A and B. 39
3.9 Example running time for Algorithms A and B on the same data size. 40
3.10 Example of correct entities of Algorithms A and B on the same running time. . 40
3.11 Accuracy vs. time on computing Stanford dependencies. 42
3.12 Raw datasets, features, and evaluation techniques used for our experiments. . . 48
3.13 Problem, datasets, and algorithms considered. 48

4.1 Summary of NER systems. 51
4.2 Comparison of the corpora properties [Atdağ and Labatut, 2013]. 53
4.3 Example errors that were found by a NER system. 56
4.4 NER error types. 57
4.5 Scalability evaluation. 58
4.6 Improved version for handling larger data. 58

5.1 Sample of Movies & TV reviews. 63
5.2 Algorithms that had the best quality, speed, and performance on 1.2 GB data size. 71
5.3 Samples of books reviews. 73
5.4 Number of ratings per category before and after removing null reviews. 73
5.5 Algorithms that had the best quality, speed, and performance on 3.3 GB data size. 78

6.1 Selected topics and article counts. 85
6.2 Algorithms that had the best quality, speed, and performance of all algorithms on 1.2 and 1.5 GB. 91
6.3 Data structure for published document. 92

6.4 Data structure for patent. 93
6.5 Target value preparation. 94
6.6 Algorithms that had the best quality, speed, and performance on 7.18 GB data size. 99

7.1 Summary of problems, final datasets, evaluation metrics, and features. 102
7.2 Dominant algorithm and best performance for all problems in the largest data size. 102

Chapter 1

INTRODUCTION

1.1 Motivation

In recent years, big data has been one of the hottest topics in computer science. The volume of data is expanding at a considerable rate with the use of technology such as smartphones, megapixel cameras, tablets, computers, ubiquitous social media, and satellites that are all generating more data than ever [Chen and Zhang, 2014, Mayer-Schönberger and Cukier, 2013, Sagiroglu and Sinanc, 2013].

Big data is commonly characterized using several V's [Sagiroglu and Sinanc, 2013]. The first three V's are Volume, Velocity, and Variety. It is these three main characterizations that define and describe the challenges that come with large data [Tarekegn and Munaye, 2016]. We describe all V's below:

- Volume describes the large amounts of data that are processed in our digital world
- Variety describes the different forms data can come in such as text, images, voice, and geospatial data. There are varying sizes of data in a variety of formats and quality which must be processed quickly.
- Velocity describes the speed at which data is processed.
- Veracity refers to the noise, biases, and abnormality in data.
- Valence describes large amounts of data in terms of graphs.
- Value refers to the real impact of the solution.

"The amount of data has been increasing and data set analyzing has become more competitive. The challenge is not only to collect and manage vast volumes and different types of data, but also to extract meaningful value from this data" [Bakshi, 2012, Nguyen et al., 2016] Without innovative, perceptive and analytical managers, it is difficult to decide how to handle large amounts of data.

The high volume of data makes extracting information a very big challenge that is both difficult and time consuming. Machine learning (ML) is a powerful tool that can help us with that task (extracting information or knowledge discovery from data). Depending on the task, we need to decide which approach is the most appropriate, such as classification, regression, or clustering. Each approach has been implemented using different ML algorithms. On the other hand, the ability of a machine learning model to perform accurately on new, unseen

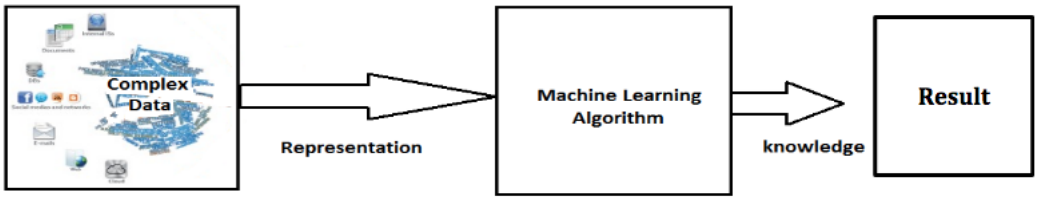

Figure 1.1: Machine learning workflow process.

examples/tasks (after being trained by a given data set) is different. Therefore, it is necessary to compare and analyze the result of different models to be able to choose the best possible for each task.

Each data set comes in one or more varieties such as text, images, voice or geospatial data. This data needs to be formatted in a specific way for a specific target to make it readable by the ML algorithm. Then, given a task, ML will teach itself to extract the correct information from the large set of formatted data to give a result. Knowledge is extracted from the results that are obtained by the ML algorithm. Figure 1.1 shows the typical workflow of this process.

Knowledge Discovery (KD) as a process, depicted in Figure 1.2, is perceived as the leading research model in academic research per Fayyad. KD consists of following steps [Cios et al., 2007]:

1. **Developing and understanding the application domain.** In this step, we learn the compatibility of previously gained insights and planned objectives of the end user of found knowledge.

2. **Creating a target data set.** In this step the data miner is selecting a subset of variables or attributes and performing a discovery task by using a data sample. In this step, the data miner usually uses queries on the existing data to select desired subset.

3. **Data cleaning and pre-processing.** This step consists of dealing with missing values and noise, removing outliers inside the data, and the consideration for the time sequence information and known changes.

4. **Data representation:** data representation contains the following two steps:

   (a) **Data reduction and projection.** This step includes finding helpful attributes by using the transformation method and dimension reduction, and finding similar representation of the data.

   (b) **Choosing the data mining task.** In this step the data miner can match the goal that is defined in Step one with a precise data mining method, like regression, clustering, classification, etc.

5. **Choosing the ML algorithm.** In this section the data miner chooses a method to find patterns in the data and find the most appropriate model and parameter of the methods used.

6. **Machine learning.** In this step, patterns in a particular representational form are generated, like, decision trees, classification rules, trends, regression models, etc.

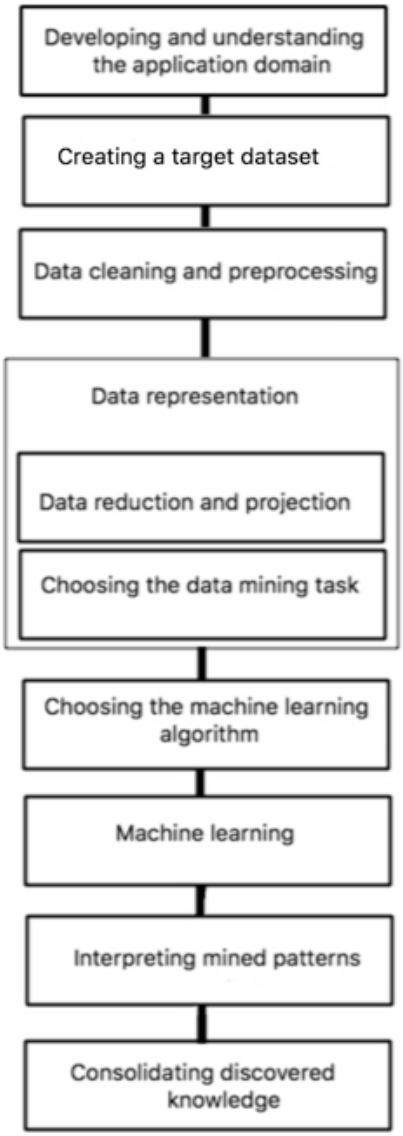

Figure 1.2: Knowledge discovery process.

7. **Interpreting mined patterns.** In this step, the analyst performs visualization of the extracted patterns and models, and the visualization of the data that is based on the extracted models.

8. **Consolidating discovered knowledge.** In the final step, we can merge the knowledge we have discovered, reporting and documenting it to the organizations that are interested. Here we may be able to check and resolve any potential conflict that may arise from any previous belief knowledge.

Our work follows mainly this process.

1.2 Formal Problem

The main goal of natural language processing and machine learning is to obtain a high level of accuracy and efficiency. Unfortunately, obtain the high accuracy often comes at the cost of

slow computation [Jiang et al., 2012]. There is a lot of research that tries to improve accuracy. There is little research, which considers time and accuracy together.

Now with increasing size of data, we will need natural language processing systems to be fast and accurate on larger amounts of data. Quick processing with accuracy of large amounts of data is just as important as being really accurate. Typically, one seeks a reasonable trade-off between speed and accuracy. "What is reasonable for one person might not be reasonable for another" [Jiang et al., 2012]. The same comment applies to a given task. We want to find an algorithm with respect to a customer-specified speed/accuracy trade-off, on a customer-specified data distribution. In most cases, it is hard to find annotated datasets and using professional humans to annotate the unannotated dataset can be expensive and time consuming.

Many algorithms exist to perform a task. Among the algorithms for a particular task, the algorithms may vary by processing methodology as well as by efficiency. This makes it difficult for a customer to select an appropriate algorithm for a specific situation. The situation is even more difficult considering that the answer may depend on the specific data set and/or its size as well as the set of algorithms and the type of evaluation used. We can even complicate even more this problem by adding space or time restrictions for the training and/or the prediction phase.

With respect to data size, when it increases, quality improves and efficiency also increases. However, after a point, quality is not increasing as much, while the running time keeps increasing and hence the quality gain is not worth it. Therefore, increasing data after that point for training is not efficient nor effective any more.

We address this problem by considering the trade-offs between efficiency and accuracy on different sizes of data and examining several algorithms on three different problems/tasks in text processing, comparing algorithms by three factors: running time, data size, and quality. To start we define a framework that allows us to compare different algorithms and define relevant trade-offs.

1.3 Contributions

The main contributions of this work are the following:

- A trade-off analysis framework between quality and efficiency that can be applied to most problems that use ML algorithms. In fact, the framework borrows from similar ideas used for generic algorithms.

- Application of this framework to text processing tasks that are typically solved with supervised ML algorithms, analyzing the impact of the object granularity of the tasks (entities, opinions, documents), the specific data set as well as the type of evaluation used (simple versus k-fold). The main finding here is that the best algorithm is not necessarily the one that achieves best quality nor the most efficient one, but the one that balances well both dimensions.

- An experimental comparison of well-known Named Entity Recognizers (SNER, INET, LingPipe) using a news dataset that is relevant on its own. The main results is that the clear winner is the Stanford NER [Liaghat, 2016].

- An experimental comparison of several ML algorithms for Sentiment Analysis using two subsets of the same dataset of reviews, to analyze what is the impact of changing the dataset when they are of similar type. For one of the subsets we also analyze the impact

of the evaluation technique used. The main result is that Support Vector Machines (SVM) is the best algorithm, followed by Logistic Regression, among the algorithms considered.

- An experimental comparison of several ML algorithms for Document Classification using two different tasks (binary and multi-class) for two different datasets (news and patents), to analyze the impact of changing those parameters. For one of the sets we also analyze the impact of the evaluation technique used. The main result is that SVM is the best algorithm, among the algorithms considered.

1.4 Organization

The outline of the rest of the thesis follows:

Chapter 2 presents the background NLP and text documents classification as well as performance trade-offs in Machine learning. Six supervised ML algorithms (Logistic regression, Decision Tree, K-Nearest Neighbors, Support Vector Machine, Naïve Bayes, and Random Forest, Conditional Random Fields Model, Multilayered Neural Networks, Hidden Markov Model, and N-gram Character Language Models) are presented. It surveys the current state of trade-off performance, speed, and sizes of data in different problems. We show how the change of data size affects the performance of models. We explain the three problems, which is our main focus and we survey some of the research on this domain.

In Chapter 3, we explore our problem statement and define the trade-off framework. We explain how measuring quality and running time affect performance. Therefore, we need to consider running time, quality, and size of data for calculating performance and comparing methods. Then, our methodology and framework are presented. This includes all the steps needed for data preparation and extraction of information from texts by machine learning algorithms. Finally, we discuss the metrics used for our evaluation.

In Chapter 4, we consider the named entity recognition problem. The dataset selected is explained and three algorithms (Stanford Named Entity recognition, Illinois NET, and LingPipe) are studied. Then our experimental results are reported.

In Chapter 5, we address sentiment analysis. We study several algorithms on two different subsets of the Amazon reviews dataset (Movies & TV and Books). Both sets are explained. All steps we followed for sentiment analysis are explained. At the end of the chapter, the results of both datasets are presented and discussed.

In Chapter 6, we address the document classification problems in two different datasets (news and patents). In the case of news, we have a multi-class prediction problem and in the case of patents we have a binary classification problem. To solve these two cases, we use up to six classification algorithms (Logistic regression, Decision Tree, K-Nearest Neighbors, Support Vector Machine, Naïve Bayes, and Random Forest).

In Chapter 7, we discuss the conclusions of this dissertation and revisit our hypothesis as posed in the formal problem section. We also propose several avenues for further research suggested by our findings.

Chapter 2

BACKGROUND

2.1 Introduction

There are a lot of problems in the world of Language Technology in general and Natural Language Processing (NLP) in particular, as shown in Figure 2.1 [Jurafsky and Martin, 2000], such as Spam detection, Part of speech tagging, Name Entity Recognition (NER), Sentiment analysis, Coreference resolution, Word sense disambiguation, Parsing, Machine translation (MT), Information extraction (IE), Question Answering (QA), Paraphrase, Summarization, and Dialog [Moens, 2006]. Also, a lot research has been done to improve accuracy, time complexity, and the effect of increasing the size of data on performance. This chapter contains reviews of some of the articles about NLP and IE problems. We reviewed studies performed on the effect of size of training data on classifier performance and investigated the trade-off between efficiency and quality. After that, we chose three of the problems where machine learning is used on text processing and reviewed the state of the art on those problems.

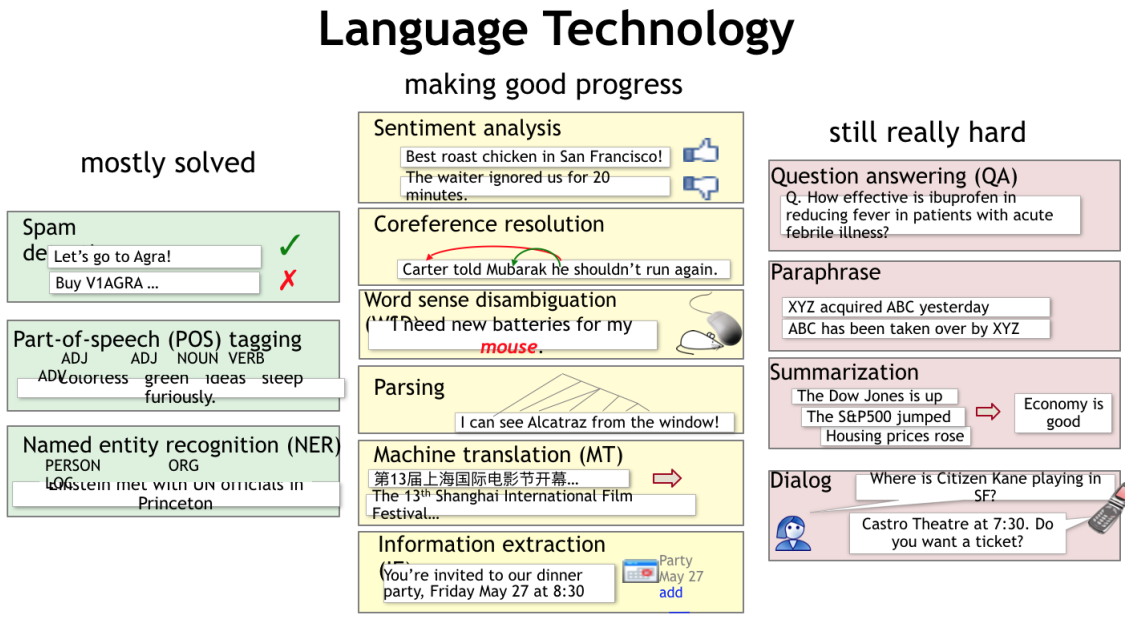

Figure 2.1: Complexity of language technology problems [Jurafsky and Martin, 2000].

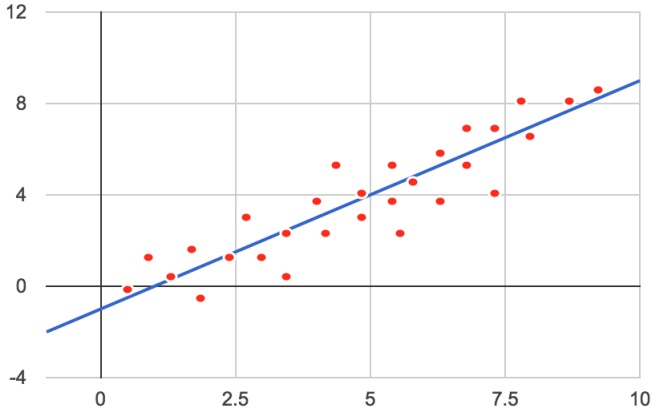

Figure 2.2: Logistic regression example.

2.2 Supervised Machine Learning Algorithms

"Machine learning is the subfield of computer science that have been developed from the fields of pattern recognition, artificial intelligence, and computational learning theory" [Isoni, 2016]. Machine learning gives computers the ability to automatically learn to make accurate predictions based on past observations. The predictions can be binary (decide between two classes) or multi-class. ML algorithms can be supervised (that is, they learn with training data) or unsupervised and their output is a model that will be used to predict. In this section, we cover the most well know supervised ML algorithms that are used in this thesis.

2.2.1 Logistic Regression

Logistic Regression uses a logistic function to measure the relationship between a dependent variable and one or more independent variable by estimating probabilities. The use of a logistic function makes logistic regression usable in many cases because it can take any positive or negative value and outputs a value between zero and one. Therefore logistic regression can be considered a probability. Figure 2.2 shows an example where the logistic function is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{2.1}$$

2.2.2 Decision Trees

The decision tree handles the classification task through creating a tree of true or false questions. The tree structure of the decision tree is well defined. The leaves represent the categories required to classify against. The nodes are the true or false queries. The root node is the document to be classified. The document passes through the queries which directs the document to one of the leaves which represent the category or the goal of classification of document. An example is shown in Figure 2.3.

The decision tree classification algorithms perform extremely well and are highly recommended in multiple cases because they have several advantages. Decision trees are easily un-

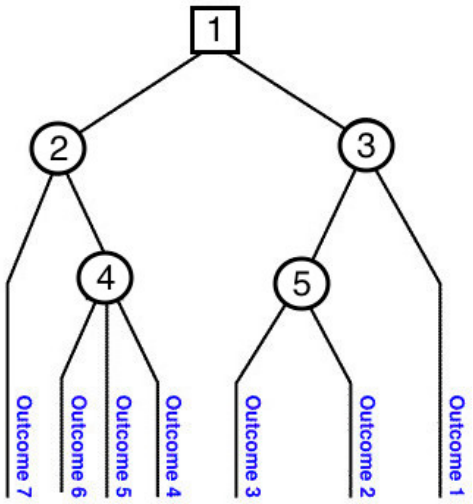

Figure 2.3: Decision trees example.

derstood and represented even to non-experts because of their simple structure. The path of the document from the root node to the leaf which shows the chosen category shows the effect of the features of the decision making process. The result of a decision tree can be replicated using simple mathematics techniques. "When there are a small number of structured attributes, the performance, simplicity and understandability of decision trees for content-based models are all advantages" [Kobsa, 2007, Khan et al., 2010].

The main disadvantage of decision trees is that they can easily overfit. This is because classification algorithms tend to classify the training data better and neglect the effect of unseen data. If the document has many features, this will lead to a complex tree structure.

2.2.3 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm measures the similarity between objects by maintaining the k-nearest neighbors to each object. Through this similarity we can determine the category of the tested object. In KNN, the categorized object is based on the nearest feature space and therefore KNN is an instance based learning algorithm.

To find the similarity between objects, the features of the training set are mapped into

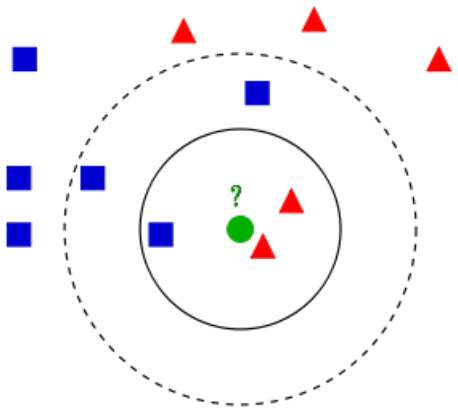

Figure 2.4: K-nearest neighbors example [Wikipedia, 2017a].

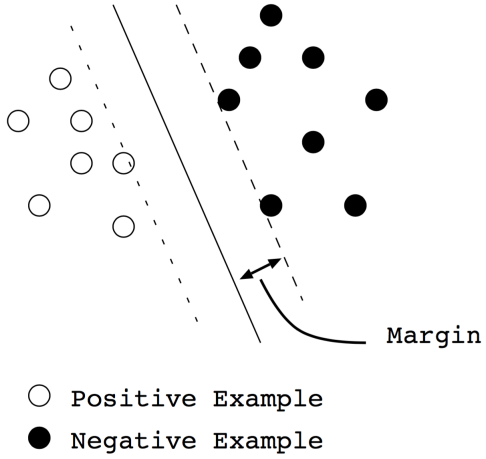

Figure 2.5: Support vector machine example [Wikipedia, 2017b].

multidimensional space. The required categories divide this multidimensional space. Each point in this multidimensional space is classified to a certain category because most of its k nearest neighbors belongs to this category. Multiple distance metrics can be used to compute the distance between a point and its neighbors. Euclidean distance is one of the most used distance metrics. See an example in Figure 2.4.

In the training phase, KNN stores the feature vectors and categories of the training set are stored. The vector of the new object is computed in the classification phase [Khan et al., 2010]. Then the distance between this vector and all the other vectors is calculated. The k smallest distances are selected and the major category from these k vectors is chosen for the object.

2.2.4 Support Vector Machines

Support Vector Machines (SVM) is considered one of the most accurate classification methods. SVM is one of the discriminative classification methods based on the idea of finding a hypothesis that finds the lowest true error. SVM differs from many classification methods in that it needs both, positive and negative examples in the training set [Khan et al., 2010].

"These positive and negative examples are needed to seek for the surface that best separates the positive from the negative data in the n-dimensional space, so called hyper-plane. The objects representatives which are closest to the decision surface are called the support vector. The performance of the SVM classification remains unchanged if objects that do not belong to the support vectors are removed from the training data set" [Khan et al., 2010]. See an example in Figure 2.5.

The SVM classification method stands out from the other methods due to its outstanding classification effectiveness. Furthermore, it can handle objects in a high-dimensional input space, and culls out most of the irrelevant features. However, the major drawback of the SVM is their relatively complex training and the high time and memory consumptions during the training stage and the classifying stage [Khan et al., 2010]. Besides, confusions may occur during the classification tasks if the documents could be annotated to several categories because the similarity is typically calculated individually for each category. So SVM is a supervised learning method for classification to find out the linear separating hyper-plane which maximizes the margin between the two datasets. "To calculate the margin, two parallel hyper-planes are constructed, one on each side of the separating hyper-plane, which are "pushed up against" the

two data sets. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the neighboring data points of both classes, since in general the larger the margin the lower the generalization error of the classifier" [Khan et al., 2010].

2.2.5 Naïve Bayes Classifier

The Naïve Bayes classifier is based on Bayes theorem of conditional probability:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \tag{2.2}$$

So the Naïve Bayes classifier is a simple probabilistic classifier that has strong independence assumptions as in Bayes theorem.

In Naïve Bayes classifiers, the feature order is irrelevant which means that the presence of a feature does not influence the other features in the classification process. This is because of the independence assumptions. Hence, the computation of the Bayesian classification is more efficient but has very limited applicability. Due to the nature of the probabilistic model, Naïve Bayes classifiers require a small set of practice data to calculate the parameters required to efficiently classify the objects with the following formula:

$$\hat{y} = \underset{k\in\{1,....K\}}{\arg\max}\, p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{2.3}$$

Naïve Bayes classifiers have been reported to work efficiently on many complex real world classification problems due to their over simplified assumptions. Naïve Bayes classification correctly classifies the objects if the correct category is more probable than all other categories. The classifier is strong enough to ignore the problems of its Naïve probability. This means that the category probabilities do not have to be calculated extremely well. The only problem with Naïve Bayes classifiers is that they are not as discriminative as some other classifiers like SVM [Khan et al., 2010].

For many years, Naïve Bayes was one of the popular machine learning methods. Its simplicity makes it attractive in various tasks and reasonable performances are obtained, although is based on an unrealistic independence assumption [Khan et al., 2010].

2.2.6 Random Forest

Random forest (RF) is an ensemble learning method for regression, classification, and other tasks, that works by building a a lot of decision trees at training time and outputting the class that is the mean prediction of the individual trees. Random decision forests corrects decision trees' usual overfitting of their training set [Fang and Zhan, 2015, Chen et al., 2004].

A random forest grows many classification trees. To classify a new item from an input vector, we put the input vector down each of the trees in the forest. Each tree gives a classification result, and the trees "votes" for that class. The forest selects the classification having the maximum votes over all the trees in the forest [Breiman and Cutler, 2016].

Each tree is grown as follows [Breiman and Cutler, 2016]:

1. If the number of instances in the training set is N, sample N instance at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

2. If there are M input variables, a number $m << M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

3. Each tree is grown to the largest expanse possible without pruning.

The random forest error rate depends on two things [Breiman and Cutler, 2016]:

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.

- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Two parameters are important in the Random forest algorithm [Fang and Zhan, 2015, Chen et al., 2004]:

1. Number of trees used in the forest (ntree) and

2. Number of random variables used in each tree (mtry).

Reducing mtry (Number of random variables used in each tree) reduces both the correlation and the strength. Increasing mtry increases both. Somewhere in between is an "optimal" range of mtry - usually quite wide. Using the Out-of-Bag (OOB) error rate a value of m in the range can quickly be found. This is the only adjustable parameter to which RF is somewhat sensitive [Breiman and Cutler, 2016].

The first step is to set the mtry to the default value (square root of total number of all predictors) and search for the optimal ntree value. To find the number of trees that correspond to a stable classifier, Random forest is built with different increasing ntree values. Ten RF classifiers are built for each ntree value, record the OOB error rate and see the number of trees where the out of bag error rate stabilizes and reaches a minimum [Bhalla, 2016].

There are two ways to find the optimal mtry [Bhalla, 2016]:

1. Apply a alike procedure such that Random forest is run 10 times. The optimal number of predictors selected for split is selected for which out of bag error rate stabilizes and reaches a Minimum.

2. Experiment with including the (square root of total number of all predictors), (half of this square root value), and (twice of the square root value). Then check which mtry returns the maximum area under curve. Thus, for $1,000$ predictors, the number of predictors to select for each node would be 16, 32, and 64 predictors.

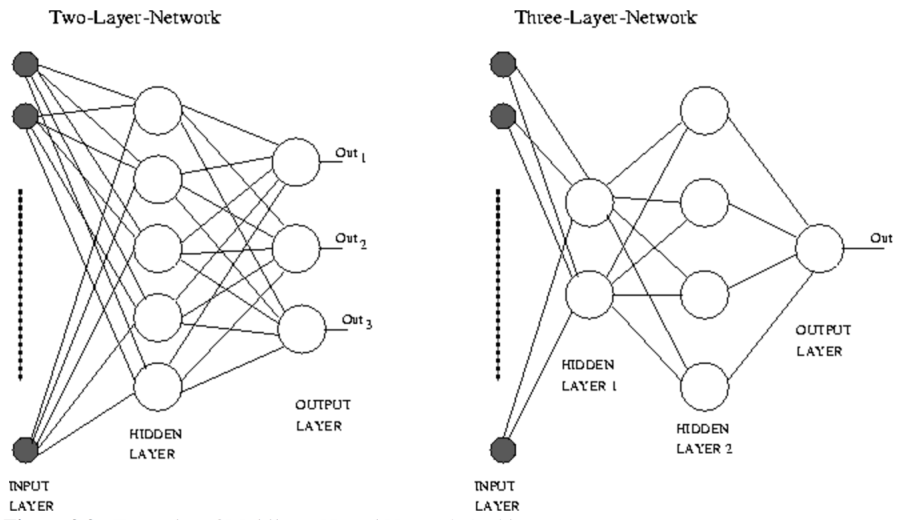

Figure 2.6: Examples of multilayer neural network architectures [Schmidt and Okt, 2000].

2.2.7 Conditional Random Fields

Conditional Random Fields (CRF) [McCallum and Li, 2003a, Finkel et al., 2005] is an undirected graphical model, which allows the bi-directional flow of probabilistic information across a sequence and provides discriminative training. CRF encodes known relationships between constructed consistent interpretations and observations. Given some observation, CRF is known as a conditional model that represents the probability of a hidden state sequence. It corresponds to conditionally-trained finite state machines. Given the values assigned to other designated input nodes, CRF can calculate the conditional probability of values on designated output nodes. In some cases, edges in a linear chain, links the output nodes of the graphical model, which is based on the same exponential form as maximum entropy models. They also have adequate methods for complete, non-greedy finite-state inference and training. It also does not consider that features are self sufficient. On the other hand, CRF is a bit slower than Hidden Markov Models (HMMs) and MaxEnt Markov Models (MEMMs).

2.2.8 Multilayered Neural Networks

A Multilayered Neural network is like an artificial neural network as it contains two or more trainable layers. Multilayer neural network is a feedforward neural network. “Multilayer Neural Networks implement linear discriminants in a space where the inputs have been mapped non-linearly. The form of the non-linearity can be learned from simple algorithms on training data” [Jeff Robble, 2008]. Several different algorithms can be used for training multilayer neural networks such as generalized delta rule and back propagation algorithms. One of the difficulties developing neural network is adjusting the complexity of the network or regularization. Multilayer neural networks can be used for nonlinear sets by employing hidden layers, whose neurons are not connected to the output directly [Schmidt and Okt, 2000]. Figure 2.6 shows examples of typical multilayer network architectures.

2.2.9 Hidden Markov Models

An HMM [Sutton and McCallum, 2006] models a sequence of observations $X = \{x_t\}_{t=1}^{T}$ by assuming that there is an underlying sequence of states $Y = \{y_t\}_{t=1}^{T}$ drawn from a finite state set S. In NLP, HMMs have been used for sequence labeling tasks such as part-of-speech tagging, named-entity recognition, and information extraction. For example, the named-entity in each observation x_t is the identity of the word at position t, and each state y_t is labeled for each word that can be identified as one of the entity types such as Location, Person, Organization, and Other. To model the joint distribution $p(y, x)$ in tractably way, an HMM makes two independence observation. In the first step, it considers each state depends only on its immediate predecessor, which explains that each state y_t is independent of all its ancestors $y_1, y_2, ..., y_{t=2}$ given its previous state y_{t-1} . Next, an HMM assumes that each observation variable x_t depends only on the present state y_t . Then, we can specify an HMM using three probability distributions: first, the distribution $p(y_1)$ over initial states; second, the transition distribution $p(y_t|y_{t=1})$; and third, the observation distribution $p(x_t|y_t)$. That is, the joint probability of a state sequence y and an observation sequence x factorizes as

$$p(y, x) = \prod_{t=1}^{T} p(y_t|Y_t - 1)p(x_t|y_t) \ .$$

2.2.10 Language Models

Language models estimate the likelihood of a given word in a text. They are trained using samples of text and then can match other text that is similar to the one used during the training phase. They use an n-gram model of sequences, notably for natural language, using the statistical properties of n-grams [Carpenter, 2013] . Hence, a n-gram model is some sort of probabilistic language model for finding the next item in a sequence of length n. The probability of a word w given some previous documents d, is computed by $P(w|d)$. In a simple example, if we want to calculate the probability that the word "the" comes after the sentence "This water is so transparent that", we can count how many times we see this sentence in all the documents, and how may times we see the sentence followed by "the" at the end of sentence. Hence:

$$P(the|its\ water\ is\ so\ transparent\ that) = \frac{Count(its\ water\ is\ so\ transparent\ that\ the)}{Count(its\ water\ is\ so\ transparent\ that)}$$

2.3 Classical Document Features

In this section we present the main features used for represent documents as vectors. This is used later in Chapter 6.

2.3.1 Term Frequency

For term frequency $tf(t, d)$ (TF), we measure the frequency of the term t in the document d [Baeza-Yates and Ribeiro-Neto, 2011]. By frequency we mean the number of occurrences of this term in this document. If frequency is $f_{t,d}$ then $tf(t, d) = f_{t,d}$. There are other ways to calculate the frequency as follows:

- Boolean: $tf_{(t,d)} = 1$ if it occurs in d and 0 otherwise;

Document 1	The night is black.
Document 2	The stars are shiny.

Table 2.1: Training dataset.

Document 3	The stars in the night are shiny.
Document 4	I can see the bright stars, the shiny stars.

Table 2.2: Test dataset.

- Logarithmically scaled frequency to smooth it: $tf_{(t,d)} = 1 + \log f_{t,d}$, or zero if $f_{t,d}$ is zero; [Baeza-Yates and Ribeiro-Neto, 2011]
- Normalized frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document [Baeza-Yates and Ribeiro-Neto, 2011]:

$$tf(t,d) = 0.5 + 0.5\frac{f_{t,d}}{\max\{f_{t,d} : t \in d\}} \tag{2.4}$$

2.3.2 Inverse Document Frequency

The inverse document frequency (IDF) measures the rarity of the term in the corpus, thus measuring how much information the word holds for the classification process. IDF is the inverse fraction of the count of documents containing the word then logarithmically scaled to smooth it. This means common words like "the" gets penalized to reduce their weight. It is defined as [Baeza-Yates and Ribeiro-Neto, 2011]:

$$IDF(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{2.5}$$

with

- N: total number of documents in the corpus $N = |D|$.
- $|\{d \in D : t \in d\}|$: number of documents where the t term appears in corpus (***i.e.*** $tf(t,d) \neq 0$). This will lead to a division-by-zero, if the term is not in the corpus. Hence, it is common to modify the denominator to $1 + |\{d \in D : t \in d\}|$..

For example [Perone, 2011], Table 2.1 and 2.2 define our document space:

Now, what we have to create a index dictionary of the words of the train dataset, we will have the following index vocabulary by using the documents one and two from the document set, define as $E(t)$ where the t is the term:

$$E(t) = \begin{cases} 1: & \text{if } t \text{ is "black"} \\ 2: & \text{if } t \text{ is "stars"} \\ 3: & \text{if } t \text{ is "shiny"} \\ 4: & \text{if } t \text{ is "night"} \end{cases}$$

Note that the terms like "are" and "the" were ignored as cited before because they do not carry any meaning to the documents. Now that we have an index dictionary, "we can convert the test document set into a vector space where each term of the vector is indexed as our index

vocabulary. So the first term of the vector represents the" "black" term of our vocabulary, the second represents "stars" and so on. Them, "we are going to use the term-frequency to represent each term in our vector space; the term-frequency is" the count of how many times the term occurs in our corpus. $E(t)$ are present in the documents tree or four, we define the term-frequency as a counting function:

$$tf(t,d) = \sum_{X \in d} fr(X,t)$$

then $tf(t,d)$ is :

$$fr(x,t) = \begin{cases} 1, \text{ if } x = t \\ 0, \text{ otherwise} \end{cases}$$

Therefore, what the $tf(t,d)$ returns the count of the occurrences of the term t in the the document d. An example of this, could be $tf(\text{starts}, d_4) = 2$ since we have only two occurrences of the term "stars" in the document d_4. Now we can create the document vector, which is represented by [Perone, 2011]:

$$\vec{v_{dn}} = (tf(t_1,d_n), tf(t_2,d_n), tf(t_3,d_n), tf(t_4,d_n), \ldots, tf(t_n,d_n))$$

The term of the vocabulary represents each dimension of the document vector. For instance, the $tf(t_1,d_2)$ represents the frequency-term of the term 1 or t_1 (which is our "black" term of the vocabulary) in document 2.

Below shows a concrete example of how the documents 3 and 4 are then represented as vectors:

$$\vec{v_{d3}} = (tf(t_1,d_3), tf(t_2,d_3), tf(t_3,d_3), tf(t_4,d_3), \ldots, tf(t_n,d_3))$$
$$\vec{v_{d4}} = (tf(t_1,d_4), tf(t_2,d_4), tf(t_3,d_4), tf(t_4,d_4), \ldots, tf(t_n,d_4))$$

This evaluates to:

$$\vec{v_{d3}} = (0,1,1,1)$$
$$\vec{v_{d4}} = (0,2,1,0)$$

Since the documents three and four are:
Document 3: The stars in the night are shiny
Document 4: I can see the bright stars, the shiny stars.

The resulting vector $\vec{v_{d3}}$ shows that we have, in order, one occurrence of the term "stars", zero occurrences of the term "black", and so on. In the vector $\vec{v_{d4}}$, we have zero occurrences of the term "black", two occurrences of the term "stars", etc.

Since we have a corpus, now represented as a vector space model, they can be represented by a matrix with $|D| * F$ shape, where $|D|$ is the total number of documents and F is the number of features (total number of different words or vocabulary). An example of the matrix representation of the vectors described above is:

$$M_{|D|*F} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

These matrices tend to be very *sparse* (with majority of terms zeroed), so they are usually represented in a compact way.

2.3.3 Term Frequency-Inverse Document Frequency

The formula for the Term Frequency-Inverse Document Frequency (TF-IDF), $TF - IDF$ is then [Baeza-Yates and Ribeiro-Neto, 2011]:

$$TF - IDF(t) = tf(t, d) * IDF(t)$$

Notice that a term gets a high $TF - IDF$ weight if the term frequency for this document is high (local context) and the inverse document frequency is low for the whole corpus (global context) .

Following our previous example [Perone, 2011], by calculating the IDF for each term (feature) present we obtain the following matrix:

$$M_{train} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

Since we have four features, we have to calculate $IDF(t_1)$, $IDF(t_2)$, $IDF(t_3)$, $IDF(t_4)$:

$$\begin{aligned} IDF(t_1) &= \log \frac{|D|}{1 + |\{d : t_1 \in d\}|} = \log \frac{2}{1} = 0.69314718 \\ IDF(t_2) &= \log \frac{|D|}{1 + |\{d : t_2 \in d\}|} = \log \frac{2}{3} = -0.40546511 \\ IDF(t_3) &= \log \frac{|D|}{1 + |\{d : t_3 \in d\}|} = \log \frac{2}{3} = -0.40546511 \\ IDF(t_4) &= \log \frac{|D|}{1 + |\{d : t_4 \in d\}|} = \log \frac{2}{2} = 0.0 \end{aligned}$$

The vector below represents the IDF weights:

$$\vec{IDF}_{train} = (0.69314718, -0.40546511, -0.40546511, 0.0)$$

After we have our matrix with the term frequency (M_{train}) and the vector representing the IDF for each feature of our matrix ($\vec{IDF}_{train}$), we can calculate our TF-IDF weights. To calculate the TF-IDF, We multiply each column of the matrix M_{train} with the corresponding $\vec{IDF}_{train}$ vector. For doing that, we need to create a square diagonal matrix called M_{IDF} with both the horizontal and vertical dimensions equal to the vector $\vec{IDF}_{train}$ dimension: [Perone, 2011]

$$M_{IDF} = \begin{bmatrix} 0.69314718 & 0 & 0 & 0 \\ 0 & -0.40546511 & 0 & 0 \\ 0 & 0 & -0.40546511 & 0 \\ 0 & 0 & 0 & 0.0 \end{bmatrix}$$

The next step is to multiply with the term frequency matrix, then the final result is:

$$M_{TF-IDF} = M_{train} * M_{IDF}$$

The matrix multiplication is not commutative, the result of $A \times B$ will not be the same as the result of the $B \times A$, and therefore M_{IDF} is on the right side of the multiplication, to

accomplish the required effect of the multiplication each IDF value to its corresponding feature [Perone, 2011]:

$$\begin{bmatrix} tf(t_1,d_1) & tf(t_2,d_1) & tf(t_3,d_1) & tf(t_4,d_1) \\ tf(t_1,d_2) & tf(t_2,d_2) & tf(t_3,d_2) & tf(t_4,d_2) \end{bmatrix} \times \begin{bmatrix} IDF(t_1) & 0 & 0 & 0 \\ 0 & IDF(t_2) & 0 & 0 \\ 0 & 0 & IDF(t_3) & 0 \\ 0 & 0 & 0 & IDF(t_4) \end{bmatrix}$$
$$= \begin{bmatrix} tf(t_1,d_1) \times IDF(t_1) & tf(t_2,d_1) \times IDF(t_2) & tf(t_3,d_1) \times IDF(t_3) & tf(t_4,d_1) \times IDF(t_4) \\ tf(t_1,d_2) \times IDF(t_1) & tf(t_2,d_2) \times IDF(t_2) & tf(t_3,d_2) \times IDF(t_3) & tf(t_4,d_2) \times IDF(t_4) \end{bmatrix}$$

An example of this multiplication is [Perone, 2011]:

$$M_{TF-IDF} = M_{train} \times M_{IDF} =$$
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0.69314718 & 0 & 0 & 0 \\ 0 & -0.40546511 & 0 & 0 \\ 0 & 0 & -0.40546511 & 0 \\ 0 & 0 & 0 & 0.0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & -0.40546511 & -0.40546511 & 0 \\ 0 & -0.81093022 & -0.40546511 & 0 \end{bmatrix}$$

Finally, we can apply our L2 normalization process to the M_{TF-IDF} matrix. This normalization is "row-wise" because we are going to handle each row of the matrix as a separated vector to be normalized, and not the matrix as a whole [Perone, 2011]:

$$M_{TF-IDF} = \frac{M_{TF-IDF}}{\|M_{TF-IDF}\|_2} = \begin{bmatrix} 0 & -0.70710678 & -0.70710678 & 0 \\ 0 & -0.89442719 & -0.4472136 & 0 \end{bmatrix}$$

The result is the normalized TF-IDF weight of our testing document set, which is a collection of unit vectors. If you take the L2-norm of each row of the matrix, you will see that they all have a L2-norm of 1 [Perone, 2011].

2.4 Performance Trade-offs in Machine Learning

There are many performance measures used in machine learning, such as [Han et al., 2011]:

- Accuracy: the percentage of test set tuples that classify correctly.
- Lift: lift is a simple correlation measure. It measures how well a targeting model can (association rule) at predict or classify cases as having an enhanced response, measured against a random choice-targeting model.
- Weighted (cost sensitive) Accuracy: It is the measure of the performance of a ML algorithm based on its accuracy of classifying a data set where the classes are imbalanced and/or the cost of errors per class is not equal. It represents the differing cost of each type of misclassification.
- Precision/Recall: precision and recall are explained in section 3.2.2.
  - F-measure: This is explained in section 3.2.3.

Author	Naïve Bayes	KNN	SVM	Neural Network
Yang	71.5	85.0	85.0	82
Weiss	**73.4**	**86.3**	**86.3**	-
Joachims	72.0	82.3	86.0	-

Table 2.3: Performance comparison of different classification algorithms [Gandhi and Prajapati, 2012].

  - Break Even Point: Tradition is to evaluate classifiers in terms of precision and recall. Breakeven is the point where precision is the same as recall.

- ROC: ROC curve is presented by plotting false positives (FP) vs. true positives (TP)
  - AUC Area: the area under the ROC curve is called AUC.

It is important to take the right measurements to evaluate our machine learning algorithms. In order to correctly measure and compare the performance of machine learning algorithms, we must choose the correct metrics. These measurements will influence the result and ultimately the decision making of which algorithm to choose [Brownlee, 2016].

Gandhi and Prajapati [Gandhi and Prajapati, 2012] described three classification algorithms and compared them with F-measure. The three classification algorithms are K-Nearest Neighbors, Naïve Bayes, and Support Vector Machines. They have researched some problems of automatically classifying text documents into categories, which are dependent on standard machine learning algorithms. They believe that the demand for text classification is increasing to a large extent. Keeping this demand in mind, there are developments of new and updated techniques happening for automated text classification for which they present an algorithm. Finally, they describe the performance of their experiment on the data by defining the settings of their scenario. Table 2.3 shows the result of the comparison of three algorithms on Reuters-21578 by break even point. Between the algorithms, which were suggested for the use in text classification, the most important one is Support Vector Machines which was shown to continuously outperform other techniques.

In another research, Cer, Marneffe, and Jurafsky [Cer et al., 2010] studies different approaches to generating Stanford Dependencies (SD), a semantically-oriented set of dependencies. SD presents the grammatical relation between words in a sentence. They study the tradeoffs between time and accuracy and examine the comparison between constituent parsers and fast algorithms that have been particularly developed for dependency parsing. Afterward, they compare these dependency parsers to techniques used to speed up the traditional ways we extract data, namely more aggressive elimination of fundamental parsers. They present different approaches in terms of aggregate speed and accuracy, and then provide explanations of characteristic errors. For their experiments, they used dual CPU Intel Xeon E5520 and Penn Treebank dataset. They compared five popular algorithms specifically designed for dependency parsing: Stanford, Charniak, Charniak-Johnson, Bikel, and Berkeley. Such parsers differ in terms of accuracy, as well as how quickly they process with respect to balancing time and accuracy. They also compare different dependency parsers: several models from Nivre, Nivre Eager, Covington, Eisner and the Re lExparser. As Table 2.4 shows, the most accurate algorithm for generating dependencies was the Charniak-Johnson re-ranking parser and the fastest algorithms were Nivre, Nivre Eager, and Covington. By using multiple threads the speed of the algorithm can be improved. Parsing speed nearly doubles when two threads are used instead of one. However, increasing to four threads results is much slower performance than just using one thread.

Parser	**Attachment F1** Unlabeled	**Attachment F1** Labeled	**Total Time** (min:seconds)
Berkeley	90.5	87.9	10:14
Bikel	88.7	85.3	29:57
Charniak	90.5	87.8	12:10
CJ	**91.7**	**89.1**	11:18
Covington	80.0	76.6	0:16
Nivre Eager	80.1	76.2	0:16
Nivre	80.2	76.3	**0:15**
Nivre Eager Feature Interact	84.8	81.1	3:23
MSTParser	82.6	78.8	6:01
RelEx	57.8	48.1	31:38
Stanford	87.2	84.2	11:05

Table 2.4: Unlabeled and labelled attachment F1 score and time "to generate standard Stanford dependencies with different types of parsers" [Cer et al., 2010].

Stanford dependencies	**Accuracy**	**Tokens per second**
Stanford english PCFG	**90.06**	123.63
Huang	90.90	**616.55**
MSTParser	91.24	239.6
Full TurboParser	92.29	209.98
Stanford RNN	93.11	66.57
Berkeley	93.33	200.0
Charniak-Johnson	93.91	100.72

Table 2.5: Basic SD parsing performance and running time [Kong and Smith, 2014].

Kong and Smith [Kong and Smith, 2014] conducted an experiment to compare the different methods of obtaining Stanford typed dependencies. They showed the trade-off between speed and accuracy in obtaining Stanford dependencies, as it shown in Table 2.5. They also examine the effects of input representations of this trade-off, part-of-speech tags, the novel use of a different dependency representation as input, and distributional representations of words. As the paper explained, parsing is well-known for the extensive computational impediments it creates in text analysis systems. They found that direct dependency parsing could achieve similar results to the Stanford CoreNLP pipeline at much greater speeds.

In another article, Jiang, Teichert, and Eisner [Jiang et al., 2012] studied the trade-off between accuracy and speed. The nominal goal of ML natural language processing (NLP) is to achieve high accuracy. Much dedicated research has been done to find exact or approximate speedups in a broad range of inference problems. They introduce a hybrid reinforcement learning algorithm that even with comprehensive features can automatically achieve better accuracies with much better improvement in speed over state-of-the-art baselines by connecting reinforcement learning and more improved learning techniques. They can also create a learning algorithm that can find better ways to balance constituency parsing. This article focuses on parsing. They expect the approach to transfer to prioritization under other agenda-based inference algorithms, such as in machine translation. They present a simple formula ($Quality = Accuracy - \lambda Time$) to obtain quality from speed and accuracy. The results of the

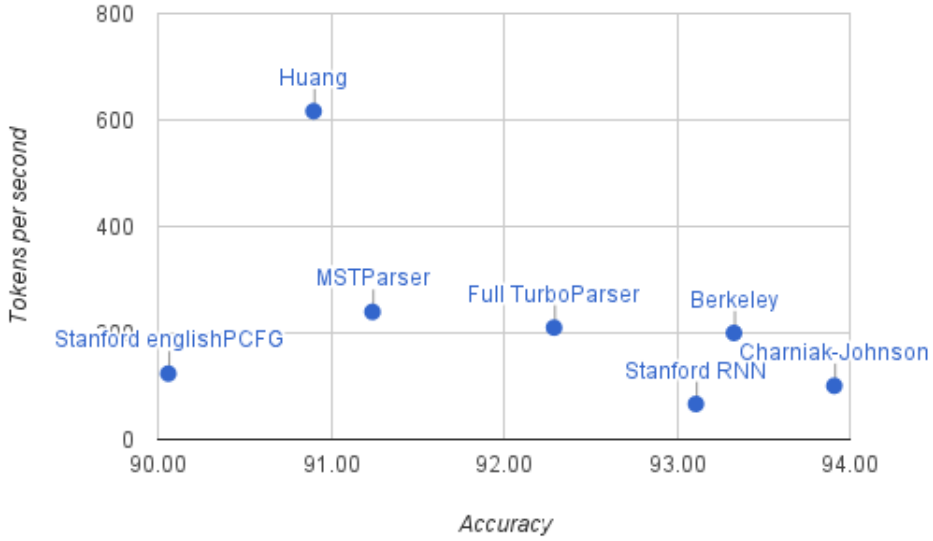

Figure 2.7: Accuracy vs. speed for SD parsing.

algorithms are compared with this formula with different factors.

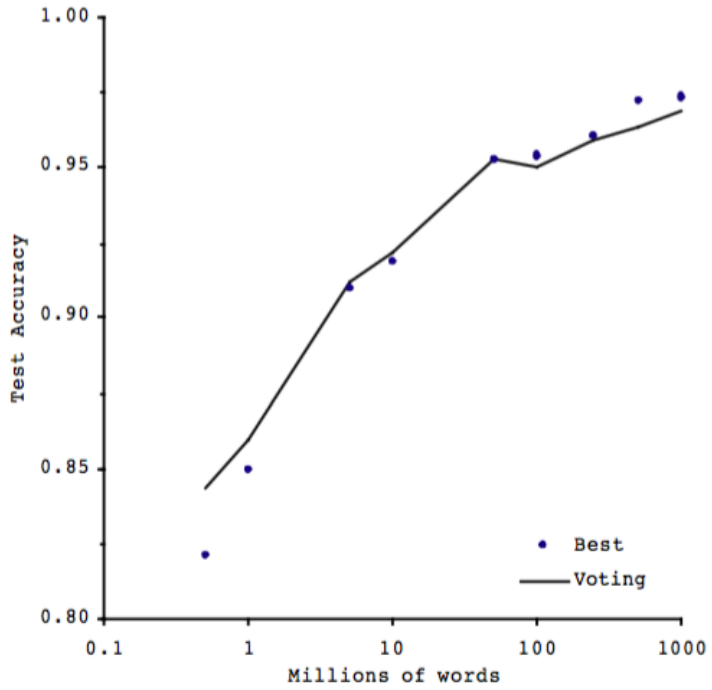

Figure 2.8: Voting among classifiers [Banko and Brill, 2001].

Banko and Brill [Banko and Brill, 2001] studied the effects of data size on ML for natural language disambiguation. They also studied the problem of selection among confusable words. The training corpus used in this study is three times larger than the training corpus which was used previously for this problem. They cover a larger size of data, and they get better results as shown in Figure 2.8. Figure 2.9 shows learning curves for four different ML algorithms. Then, they examine the efficiency of voting, selection of samples, and partially unsupervised learning with large training corpora with respect to being able to retrieve the benefits that may come from much greater training corpora without acquiring greater cost. One billion words of data were collected from a mixture of English texts. These include scientific abstracts, news articles, literature, government transcripts, and other texts. In this article, the effect of data size on machine learning for classification is studied.

In another paper, Banko and Brill [Banko and Brill, 2001] applied machine learning techniques to the task of confusion set disambiguation. They used more than a thousand times more data than previously researchers had been used for their training in disambiguation-in-string-context problem. They collected a one billion word training corpora from news, government, articles, scientific abstract, etc. They try to find out effect of training data size on performance and when benefit from additional training data will cease in the learning methods in Natural language processing. Also, they analyzed residual errors made by learners when issues of sparse data have been significantly mitigated. They reduced the error rate, and they compared performance the best system trained on the standard training set size, by adding more data. Figure 2.10 shows benefit from additional training data.

Ma and Ji [Ma and Ji, 1999] reviewed various general techniques on supervised learning to improve performance and efficiency. They introduce performance as “the generalization capability of a learning machine on randomly chosen samples that are not included in a training set. Efficiency deals with the complexity of a learning machine in both space and time”. This paper presents two factors, bias and variance dilemma, as important factors that effects

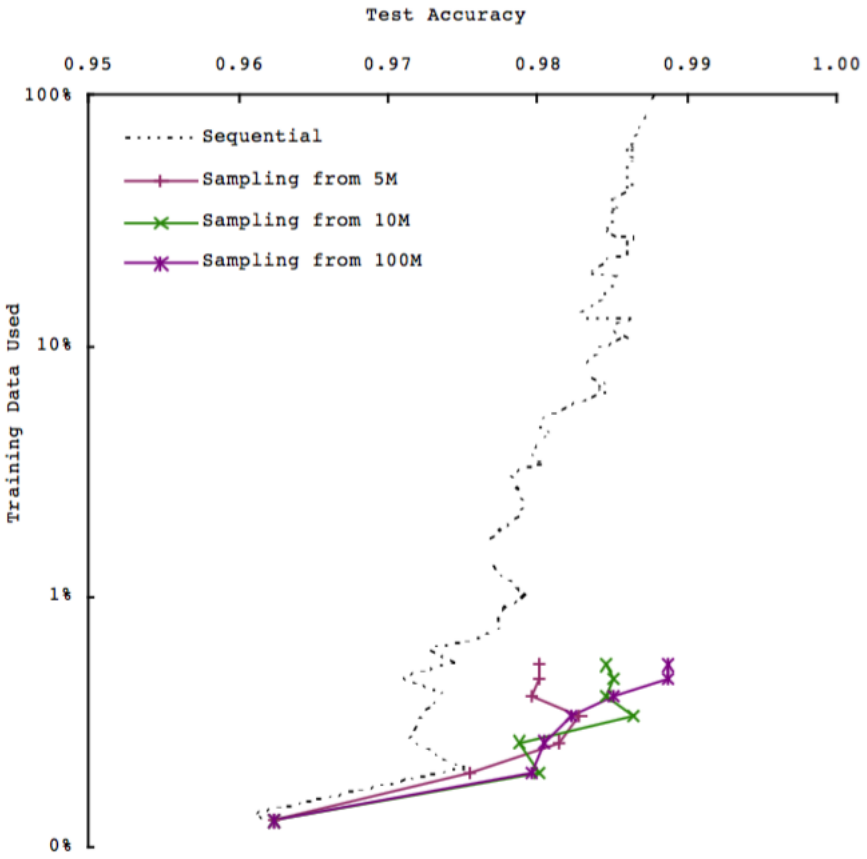

Figure 2.9: Active learning with large corpora [Banko and Brill, 2001].

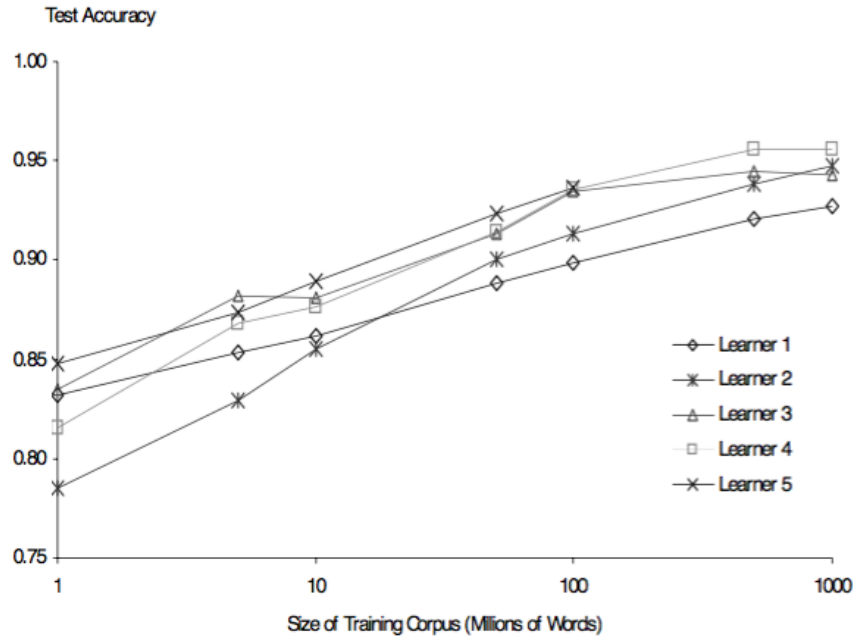

Figure 2.10: Learning Curves for Confusable Disambiguation [Banko and Brill, 2001].

performance. The authors present and discuss three types of learning approaches: training an individual mode, combinations of many weak models, and combinations of several well-

trained models. For their experience Decision tree, Incremental Learning, and Em algorithm implemented. Three different approaches to improve the performance have been discussed through making a trade-off between the variance and bias evolutionary computation of models: 1-finding the an optimal structure for a single model, 2-training several oversized models which have a high variance and low bias and then decrease the variance by combining well-trained models, 3- training a large set of weak models which have a small variance and a large bias and then decrease the bias by combinations of weak classifiers. They also evaluate the advantages and disadvantages of each approach.

In the next sections, we describe the three problems we study (Named entity recognition, Sentiment analysis, Document classification) and then review the state of the art for each problem.

2.5 Named Entity Recognition

Named Entity Recognition (NER) technology helps to recognize proper nouns (entities) in a text while correlating them with applicable types. There are several common types in a NER system, including a person's name, date, location, address, etc. Some NER systems are integrated into Parts-of-Speech (POS) taggers; still there are many stand-alone applications. Most NER systems analyze POS tag patterns by using lists of typed entities (like list of possible locations) or regular expressions of certain types (address patterns) [Sun, 2010].

Figure 2.11 is an online demo from Stanford NER system, marked up with seven entity types: <Location>, <Organization>, <Date>, <Money>, <Person>, <Percent>, and <Time>. This system gets text, types of classifier, and output format as input. After submitting, we can see the result of the classifier in highlights. Each class uses a different color.

Machine-learned NER have been designed with many different classifiers [Finkel et al., 2005]. A type of discriminative probabilistic model that is used often is a conditional random field for labeling or parsing sequential data. For example, in natural language text or biological sequences. CRF, like Markov random fields, is an undirected graphical model where each vertex represents a random variable for which the distribution is implied and the individual edges serve as a dependency between two random variables [Sun, 2010]. Now we cover some NER applications.

Radev and Dragomir [Radev, 1998] used machine learning technology for extracting a set of rules that can predict option of description out of an entity profile. They have used $35,206$ tuples, that contain entity, a description, article ID and position of entity description occurs on the articles. A descriptions identifies a textual passage is described in a given Name Entity. For instance, Barack Obama can be described as "the President of the U.S.", or "the Democratic presidential candidate" or "a Hawaiian native", depending on the document. Personal name authorization is used as a description or identification in a cue. Description identification can be used as a cue in personal name disambiguation. Reusing these describers in a context of natural language generation was the authors' intention.

Mann, Gideon S. and Yarowsky [Mann and Yarowsky, 2003] presents personal name disambiguation as a task of identifying the correct referent of a given designator. In each context, it identifies whether Jim Clark is a film editor, Netscape founder, or even a race car driver. Corpus-wide authorizes information retrieval from personal names applications in document clustering.

Dimitrov, Bontcheva, Cunningham, and Maynard [Dimitrov, 2005] worked on Entity anaphora resolution that mainly consists of resolving pronominal co-reference when the antecedent is an

Figure 2.11: An example of NER system.

Named Entity (NE). For example, in the sentence, "John finished playing the game boy and he took it back to the store", the pronoun "he" refers to "John". Anaphora is one of the ways used in solving the NER problem which can be done by allowing the use of extended co-reference networks.

Charniak [Charniak, 2001] worked on analysis of name structure. In analysis of name structure they break down a person's name into parts. For instance, the name "Doctor Aileen A Anderson" is made of the title, first name, middle initial, and the last name. This is a way to start the process of the first step toward NER and toward the solution of co-references to determine that "George W Bush" and "President Bush" are the same person, but "George W Bush" and "Laura Bush" are two completely different people.

Swan and Allan [Swan and Allan, 1999] studies the key items in a corpus at any period or given time. The event detection application becomes broader significantly by analysis or aggregation. Extracting events on several news articles or a given scenario, Swan and Allan can create a story that is made of chosen textual passages. For example, the story is that "Obama was elected as president" on November 18th and "Obama picks Biden as Vice President" on November 25th. Commercial buzz is a simple analysis of entity frequencies over time.

To identify semantics of interest in unstructured text we use NER purports. This includes finding ways to add structure to unstructured data. Investigation has been done on NER and classification (NERC) by using supervised, semi-supervised and unsupervised learning methods. The investigation was done in special domains and multilingual settings, such as biomedicine. Ekbal, Sourjikova and Frank [Ekbal et al., 2010] evaluated the challenge of fine grained NER and classification. They introduced a method to perform fine grained NERC

on a large scale. A pattern-based approach was proposed to acquire the fine grained semantic classes and instances. The baseline of this approach is modeled using Maximum Entropy (ME). To model this approach, different features were needed. These features included: part of speech tagging, word prefix and suffix, chunk information, capitalization, word length, context words, and some dynamic features.

Smith [Smith, 2002] worked on detect the physical entities in conjunction with any other entities. For example, symposiums consist of name, location and beginning and end date, (*e.g.*, name: "Science Symposium", location: "Los Angeles", start date: "February 10th, 2017", end date: "February 13th, 2017"). Another example is a person's date of birth (name: "James Jones", date: "September 21, 1977").

Srihari and Li [Srihari and Li, 1999] studied question answering that involves NER at its core of the answering efficiency. The low-level information extraction like NER is important component to handle most types of questions. Eighty percent of 200 questions regarding a TREC-8 competition required the response of a named entity (e.g. when [time or date], who [person], where [location]).

In another paper about NER, Zhang and Pan [Zhang et al., 2004] focused on the problem of searching for the thematic named entity among all items in a document. They proposed a statistical model for important NER by converting it to classification algorithms in Machine learning. They also compare several classification algorithms such as RRM, Decision Tree and Naïve Bayes. The data set was from news articles. The features were used to identify type, In Title or Not, Document Frequency in corpus and Entity Frequency.

Nadeau and Sekine [Nadeau and Sekine, 2007] present a survey of fifteen years of research in NERC from 1991 to 2006. They introduce Named entities task that can also be looked at as an Information Extraction task. This way, the entities (company activities and defense related activities in this research) are extracted from any unstructured text like news articles. In this task, it was noticed that it is important to identify information expressions and numerical expressions. Information expressions are like names, organizations and location names. Numeric expressions are like time, data, percent and money. They also present the word level features and evaluation techniques for NERC.

Florian, Ittycheriah, Jing, and Zhang [Florian et al., 2003] present a classifier combination with NER for four different classifiers (Robust Risk Minimization, Maximum Entropy, Hidden Markov Model, and Transformation-Based Learning) and they compare the results for two languages; German and English. As a machine learning method, they introduce the RRM classifier algorithm as a good candidate algorithm for NERC.

Pasca, Lin, Bigham, and Jain [Pasca et al., 2006, Nadeau and Sekine, 2007] used techniques inspired by mutual bootstrapping. The distributional similarity to generate synonyms can be used for pattern generalization. For example, In a pattern like: [PERSON was born in December], synonyms for December are the other 11 months generating new patterns as [PERSON was born in June]. In this research, it was shown that using this technique on very large corpora (100 million web documents) a small seed of 10 examples can be used to generate 1 million facts with 88% precision. They also show that a large collection of data is not sufficient alone to train a NE classifier and that it can be improved using information retrieval relevance measures and focusing on specific contexts that enrich the classifier.

Heng and Grishman [Ji and Grishman, 2006, Nadeau and Sekine, 2007] addressed the problem of unlabeled data selection. They applied two semi-supervised learning algorithms (bootstrapping and self-training) to improve F-measure on Chinese and English language datasets. Using the bootstrapping method, they can improve an existing NE classifier. They reported that is not possible to rely on large collection of documents by itself. In order to get the best

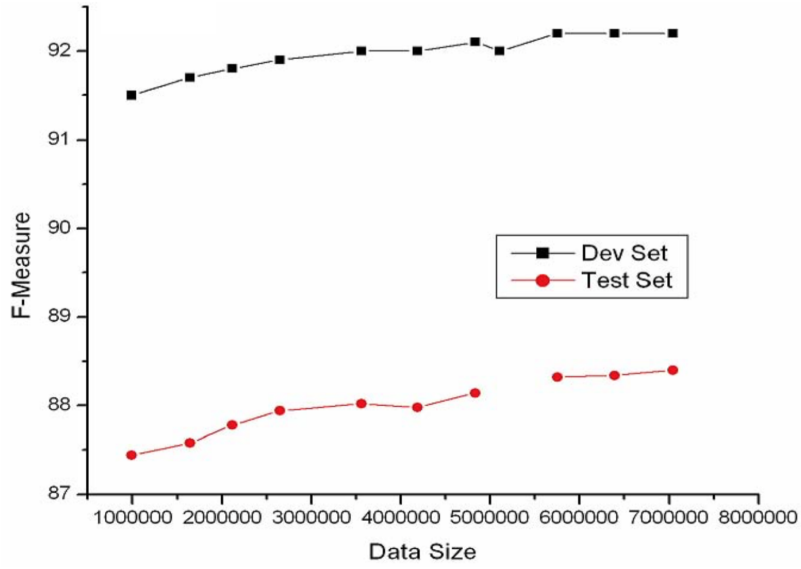

Figure 2.12: Impact of data size [Ji and Grishman, 2006].

result, they must select the documents using information retrieval-like relevance measures and specific contexts that are rich in proper names and co-references. They found performance can get worse by adding off-topic unlabeled data and it is a demand to have a good data selection measures for some application. Figure 2.12 shows the result of each segment of the unlabeled data is added to the training corpus.

2.6 Sentiment Analysis

Sentiment classification is a kind of text classification problem. Text Classification is aimed at categorizing the text to one of many topics according to the available categories. The classification is performed through the topic related words. For instance, “Politics” topic is represented by words representing different political aspects like “president” , “country”, “policy” and many more. Sentiment analysis is the analysis aimed to understand the hidden emotion, subjectivity, opinion, review, or any hidden meaning in the context. Therefore sentiment analysis can have many names, all under the same meaning such as opinion extraction, opinion mining, review mining, sentiment mining, affect analysis, emotion analysis, subjectivity analysis, etc. In sentiment classification, the key features are words that represent positive or negative opinion like: awesome, great, amazing, worse, bad, etc. [Liu, 2012].

Pang, Lee, and Vaithyanathan [Pang et al., 2002] were the first researchers to take a negative or positive opinions approach to classify online movie reviews into two classes (negative and positive). They used bigrams and unigrams as features in classification which performed quite well with either Naïve Bayes or SVM. They also tried several other characterization options as well. They implemented three machine learning methods (Naïve Bayes, Maximum entropy classification, and Support vector machines). The authors also introduced sentiment analysis as a very result driven NLP task which uses many NLP sub-tasks to give perceptive analysis from various text sources.

Qu, Ifrim and Weikum [Qu et al., 2010] introduced the “bag-of-opinions”: a certain way to represent the documents that is a representative of documents used to show the power of n-grams with opinions. The bag of opinions is a variant of the traditional bag of words where

Method	Twitter	MySpace	YouTube	BBC	Digg	Runner's World
Emoticons	**0.929**	0.952	**0.948**	0.359	**0.939**	**0.947**
Happiness Index	0.774	0.925	0.821	0.246	0.393	0.832
LIWC	0.690	0.862	0.731	0.377	0.585	0.895
PANAS-t	0.643	**0.958**	0.737	0.296	0.476	0.689
SASA	0.750	0.710	0.754	0.346	0.502	0.744
SenticNet	0.757	0.884	0.810	0.251	0.424	0.826
SentiStrength	0.843	0.915	0.894	**0.532**	0.632	0.778
SentiWordNet	0.721	0.837	0.789	0.284	0.456	0.789

Table 2.6: F-measures for the eight methods [Gonçalves et al., 2013].

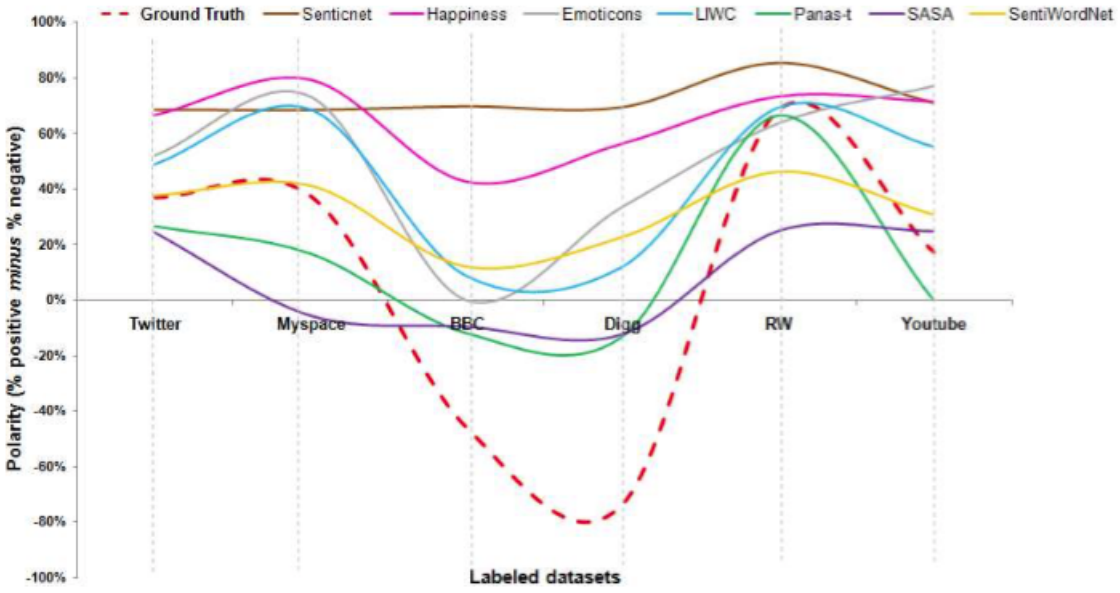

Figure 2.13: Polarity of all methods across the labelled datasets [Goncalves et al., 2013].

each opinion consists of three parts: sentiment, modifier, and negator. For instance, in "not very good", the sentiment word is "good", the modifier is "very", and the negator is "not".

Goncalves, Araújo, Benevenuto, and Cha [Gonçalves et al., 2013] investigated multiple methods for measuring sentiments, including supervised machine learning methods and lexical-based approaches. They developed a new method that is a combination of eight methods. They compared all methods on small data size from Twitter, MySpace, YouTube, BBC Forum, Runner's World, and Digg. The results for the F-measure are shown in Table 2.6 while Figure 2.13 shows the result of polarity for eight methods in different datasets.

They also computed and compared the coverage of all the methods on representative events from Twitter. As shown in Figure 2.14, the proposed method (the red point) has better quality and better coverage compared to all other methods. Figure 2.15 shows the trade-off between the coverage versus F-measure for right methods including the proposed method.

Fang and Zhan [Fang and Zhan, 2015] have investigated the issue of sentiment polarity categorization. This is one of the issues of sentiment analysis. In this paper, they were geared toward the fundamental problem of sentiment analysis and sentiment polarity categorization. They have used the product reviews from Amazon.com as data for this study. This data was selected from February to April 2014, and included, in total, over 5.1 million of product reviews

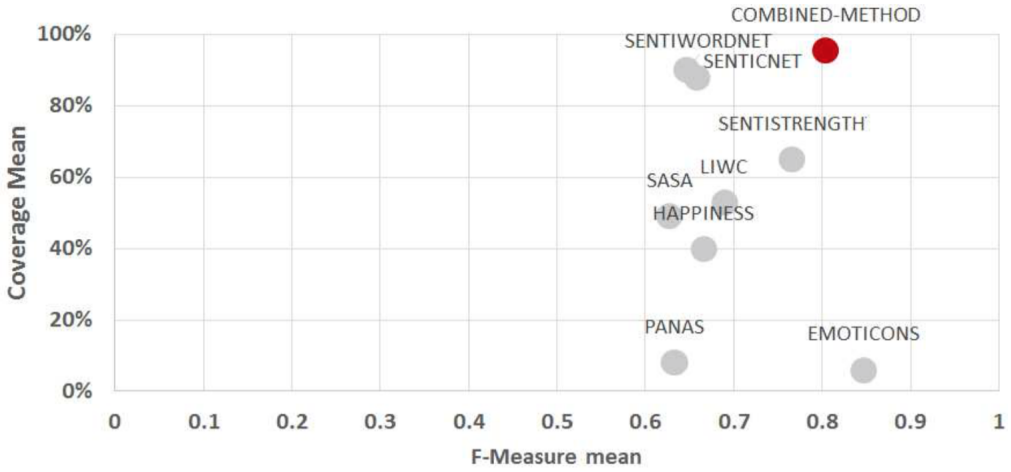

Figure 2.14: Coverage and F-measure comparison for all methods [Gonçalves et al., 2013].

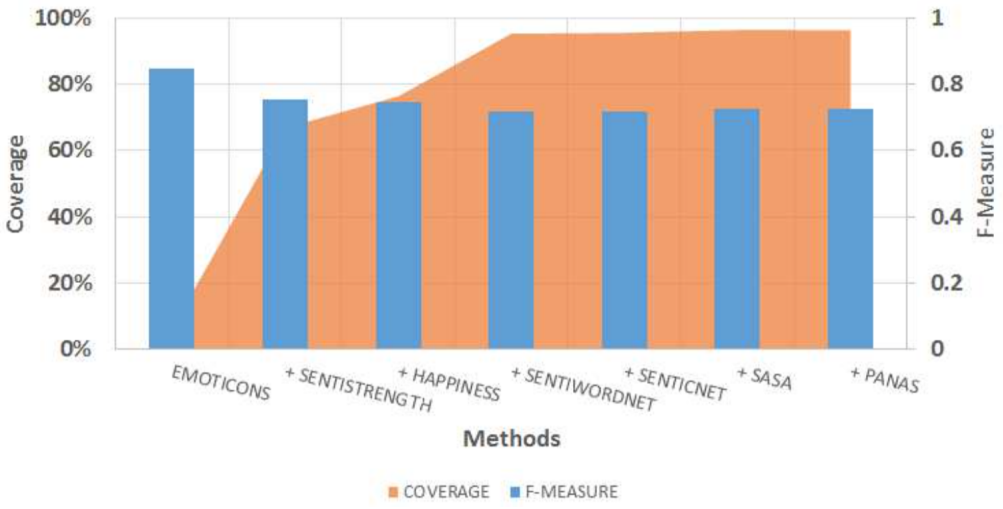

Figure 2.15: Coverage vs F-measure trade-off for all methods [Gonçalves et al., 2013].

in which the products belong to four major categories: beauty, book, electronic, and home. Data collection based on product review and category review is presented in Figures 2.16 and 2.17. "Those online reviews were posted by over 3.2 million reviewers towards 20,062 products. Every rating is based on a 5-star scale (see Figure 2.17), resulting in all the ratings ranging from 1-star to 5-star with no existence of a half-star or a quarter-star" [Fang and Zhan, 2015].

This study gives a detailed description of the steps taken along with the sentiment polarity categorization process. Three algorithms have been used to run experiments on both, review-level categorization and sentence-level categorization: Random forest, Naïve Bayes, and SVM. Figure 2.18 shows the study's score of review-level categorization.

2.7 Document Classification

Document Classification is "the task of automatically classifying a set of text documents into different categories from a predefined set" [Wajeed and Adilakshmi, 2009]. More formally,

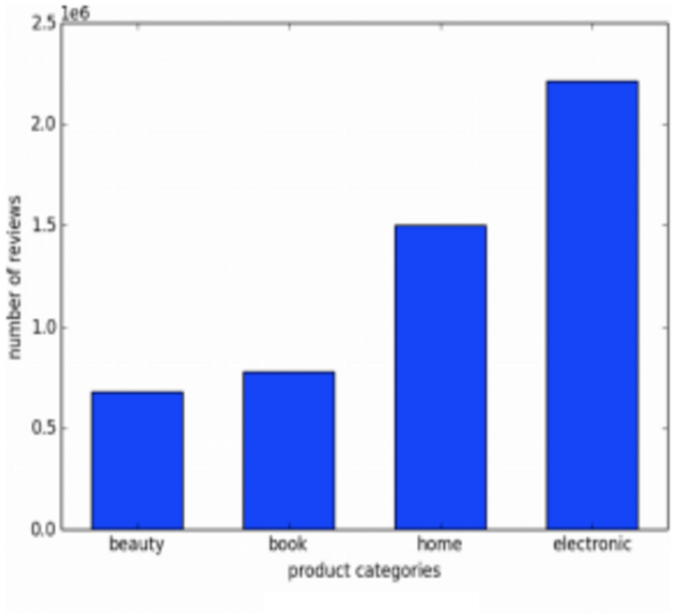

Figure 2.16: Data based on product categories [Fang and Zhan, 2015].

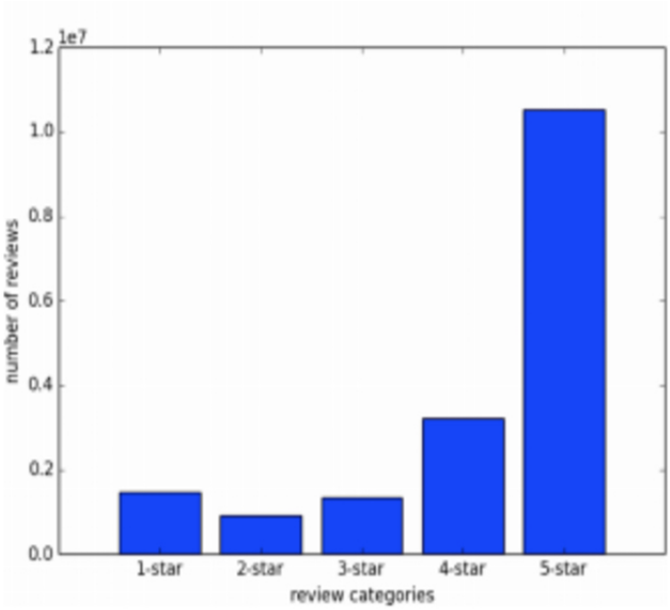

Figure 2.17: Data based on review categories [Fang and Zhan, 2015].

given a set of training documents $D = \{d_1, \ldots, d_n\}$ with known categories $C = \{c_1, \ldots, c_n\}$ and a new document d' we need to predict the category of d'. That is, the methods will associate with d' one or more of categories in C [Ikonomakis et al., 2005, Wajeed and Adilakshmi, 2009].

Machine learning and Information retrieval systems are used for text classification. Due to its importance, text classification grabbed a lot of attention from industry and researchers. They use these systems due to its content-based document management task, and because it shares many characteristics among other information retrieval tasks. Text search is an example of how to choose a set of documents that is most relevant to a particular query. For example, documents used in classification tasks are indexed using the same techniques as in IR; moreover, documents are compared and the similarity between them is measured using techniques originally developed for IR. The evaluation of classification tasks is often done using the same

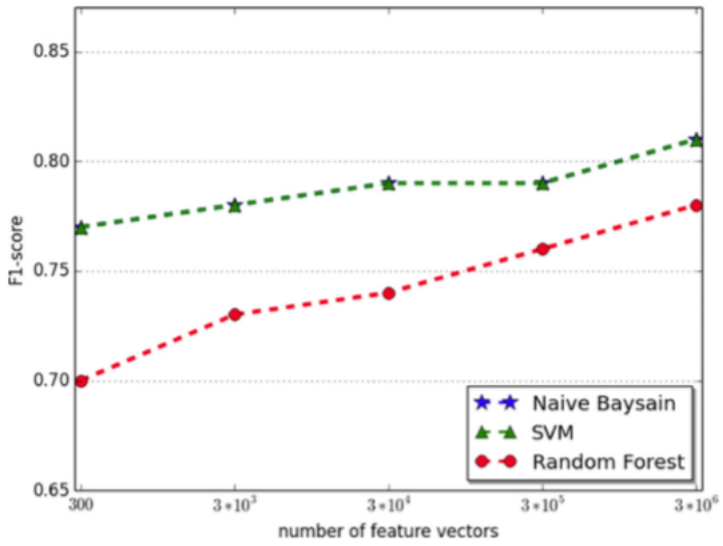

Figure 2.18: F1 Score of review-level categorization [Fang and Zhan, 2015].

effectiveness measures as in IR. For industry developers, TC is important because of the large quantity of documents that need to be properly processed and classified. Even more important is the fact that automatic TC techniques have advanced to levels of accuracy that can compete with the performance of trained professionals. TC is of interest also for ML researchers, because applications of TC are a challenging benchmark for their own techniques and methodologies. This is because TC applications use very high-dimensional feature spaces and very large amounts of data [Cachopo, 2007].

Classification tasks can be a single label classification or multi label classification depending on the application. Single label classification is when the document is classified into exactly one class, like classification of an email message to spam class or legitimate class. Multi label classification is when the document is assigned to more than one class, like a news article about "How Chris spent his vacations" can be classified to both social and politics classes [Wajeed and Adilakshmi, 2009].

We focus now on the text classification case where each document belongs to a single class, or single-label classification. Later we apply this case to the binary case (two classes) and the multi-class case.

Khan and Baharudin [Khan et al., 2010] present a challenge that comes with the increase of available electronic documents. In this research, they highlighted the important different techniques and methodologies that are used for text documents classification. At the same time, they provide there are still text representation and machine learning techniques that provide fodder for some interesting future investigation. The research focused on both text representation and different machine learning techniques through reviewing the theory and the methods of document classification and text mining.

Ikonomakis, Kotsiantis, and Tamoakas [Ikonomakis et al., 2005] studied the text classification process using machine learning techniques. It also guides through interesting research directions. They show that the classifier's performance can be improved by manipulating training text data also presented two problems in text mining: polysemy and synonymy. They express how up to now little research work in literature has been seen on how to exploit training text corpuses to improve classifier's performance. They mention some of the important tasks in the classification problem, and present two open problems in text mining: polysemy

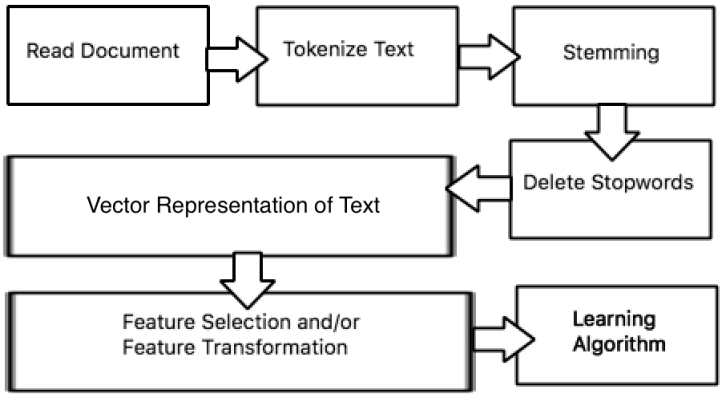

Figure 2.19: The text classification process.

and synonymy.

Figure 2.19 shows the graphical representation of the text classification process that is presented by Ikonomakis in 2005.

Wajeed and Adilakshmi [Wajeed and Adilakshmi, 2009] presented a problem of having massive amount of unstructured data and discussed how to explore and analyze this type of data. Also, it shows the types of classification structures and the ease of using hierarchical classifiers over non-hierarchical methods in the case of having a large number of documents. A major part of the paper contributes on insights towards generating the lexicon. The classifier is trained using a feature vector of the training documents. This feature vector is formed from a set of lexicons. A lexicon is the set of words the document contains. This way, the words in the document are the features of classification after passing through a preparation phase. The classifier then assigns the document to the best label.

Khan, and Bahurdin [Khan et al., 2009] overviewed how the increase in massive textual data highlights the need for text mining, machine learning (ML) and natural language processing (NLP) to extract knowledge from these documents and be able to use them efficiently. Therefore, automatic documents classification is being focused on with all its stages like document representation, features extraction, and document classifier technique. Multiple classifiers are being studied and compared to choose the best classifier for each application. The classifiers include K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM), Bayesian, Fuzzy Correlation, Artificial Neural Networks (ANN), and Genetic Algorithms. Over multiple research, SVM and Naïve Bayes are recognized as two of the most effective classifiers for textual classification which makes them very well suited for supervised machine learning classification.

Chapter 3

TRADE-OFFS FRAMEWORK

3.1 Introduction

Text processing is of great practical importance today given the massive volume of online text available. Most recently, the volume of digital text available from sources such as the World Wide Web, electronic mail, corporate databases, chat rooms, and digital libraries has exploded. Extracting any type of information is becoming a big challenge because processing high volumes of data makes it that much harder to classify out smaller more specific texts.

As quality is as important as time efficiency, we need a fair way to compare algorithms that achieve different quality at a different processing time. Most of the time the best quality algorithm is the slowest one and is not clear if the extra processing time is worth the quality improvement. Hence, we need to explore the trade-offs between quality and time in a way that is independent of the problem being solved as well as the computational infrastructure that is being used.

Hence in this chapter we first explore how to measure quality and then we propose a performance measure that combines quality and time and the notion of "dominant algorithms". Then we explain the methodology that we use in the rest of the thesis and the rationale for the problems and experiments chosen, where we apply our trade-off analysis framework.

3.2 Measuring Quality

There are several evaluation measures for predictive algorithms. Among them, we will explain the F-measure which is a gauge for quality and which we will be using in our experiments. In quality evaluation, we compare model A with model B in terms of their respective F-measure. To calculate an F-measure we need to use a confusion matrix. From that matrix, we can calculate precision and recall which are the two parameters required to calculate the F-measure. These concepts are explained below.

3.2.1 Confusion Matrix

The confusion or error matrix is a table that shows the performance of a predictive algorithm, typically used for supervised ML algorithms [Stehman, 1997]. In the confusion matrix we show the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) for the classes involved in the problem. The definition for each one of these measures is the following:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positives (TP)	False Negatives (FN)
Actual Negative	False Positives (FP)	True Negatives (TN)

Table 3.1: Contingency table binary categorization quality for a category.

Document Classification		**Predicted Class**		
		Sport	Economic	Policy
Actual Class	Sport	7	4	1
	Economic	3	5	0
	Policy	0	1	9

Table 3.2: Document classification confusion matrix.

- True Positives (TP): TP is the number of instances when the identifier's predicted result has been correctly identified and marked as the same.
- False Negatives (FN): FN is the number of instances when the identifier's predicted result has been misidentified and labeled as different when it should have been the same.
- False Positives (FP): FP is the number of instances when the identifier's predicted result has been misidentified and labeled as the same when it should have been different.
- True Negatives (TN): TN is the number of instances when the identifier's predicted result has been correctly identified and labeled as different.

Table 3.1 shows an example for the case of two classes.

Let us give a specific example. If a document classification algorithm has been trained to distinguish between sport, economic and policy documents, the results of testing will be summarized in a confusion matrix to inspect the algorithm more. Assuming a sample of 30 documents, 12 sports, 8 economics, and 10 policies, the resulting confusion matrix could look like Table 3.2:

The confusion matrix shows that from the eight articles in economics category, the system successfully predicted five to be in the economics category, but predicted wrongly three as sports. Also, from the twelve sports articles, the system successfully predicted seven to be in sports category, but predicted wrongly four as economic and one as policy. From the confusion matrix, the system can't distinguish well between sport documents and economic articles, but can make the distinction between policy documents and other types of documents well. All correct predictions are located on the diagonal of the table, making it easier to visually inspect the table for errors, as these will be represented by values outside of the diagonal. Using the confusion matrix above, the corresponding table of confusion for the sport class is shown in Table 3.3.

3.2.2 Precision and Recall

Precision and Recall are two performance measures used for evaluating the quality of results. We use precision to compute exactness, whereas recall is a measure of completeness. In the classification task, precision is defined as the ratio of correctly identified classes (TP) by the total number of instances predicted for that class whether correctly labelled or not which is the sum of the true and false positives TP and FP. As such, precision can be written down as

7 true positives (actual sports that were correctly classified as sports)	5 false negatives (sports that were incorrectly marked as Economics)
3 false positives (Economics that were incorrectly labeled as sports)	15 true negatives (all the remaining documents, correctly classified as non-sports)

Table 3.3: Table of confusion for the sport's class.

equation 3.1 and points to the number of selected items that are relevant. Recall on the other hand is the ratio of correctly identified classes (TP) by the total number of actual instances for that class correctly identified or not which is the sum of the true positives and false negatives TP and FN. As such, recall can be written as equation 3.2 and points to the number of relevant items that have been selected.

$$Precision = \frac{\#TP}{(\#TP + \#FP)} \tag{3.1}$$

$$Recall = \frac{\#TP}{(\#TP + \#FN)} \tag{3.2}$$

3.2.3 F-measure

The F-measure (F1 score) is a measure of a test's quality. The F-measure computes by both precision and recall for the test and returns a score. We can consider it as a weighted average of the precision and recall, with the F-measure reaching its worst value at 0 and best value at 1.

The traditional F-measure or balanced F-measure can be defined in more mathematical terms as a harmonic mean as shown in Equation 3.3:

$$F - measure = 2\frac{Precision * Recall}{Precision + Recall} \tag{3.3}$$

3.2.3.1 Micro and Macro Average

If one category is represented in the corpus, much more than other categories, the F-measure may not be a good way to evaluate the result. To calculate a well-representative F-measure, F-measures across all categories must be averaged. There are two types of averaging, Micro and Macro averaging. Macro averaging is averaging after calculating the F-measures for all the categories. To calculate let $tp_\lambda, fp_\lambda, tn_\lambda$, and fn_λ be the number of true positives, false positives, true negatives and false negatives where $\lambda = \{\lambda_j : j = 1 \dots q\}$ is the set of all labels and the binary evaluation measure is $B(tp_\lambda, fp_\lambda, tn_\lambda, fn_\lambda)$ [Asch, 2013]. Then

$$B_{macro} = \frac{1}{q}\sum_{\lambda=1}^{q} B(tp_\lambda, fp_\lambda, tn_\lambda, fn_\lambda) \tag{3.4}$$

That is, the Macro average is computed by first calculating the F-measures for all the categories and then taking their arithmetic mean. It works better on small-sized categories.

Micro averaging is better used on large-sized categories. It is computed by calculating precision and recall for each category first, then using them to calculate the F-measure. Micro

Sport Class		
	Truth: Positive	**Truth: Negative**
Predicted class: Positive	12	8
Predicted class: Negative	5	100

Table 3.4: Sport class.

Economic Class		
	Truth: Positive	**Truth: Negative**
Predicted class: Positive	80	10
Predicted class: Negative	30	40

Table 3.5: Economic class.

Micro Ave. Table		
	Truth: Positive	**Truth: Negative**
Predicted class: Positive	92	18
Predicted class: Negative	35	140

Table 3.6: Micro Average Table.

average is computed as follows [Asch, 2013]:

$$B_{micro} = B\left(\sum_{\lambda=1}^{q} tp_\lambda, \sum_{\lambda=1}^{q} fp_\lambda, \sum_{\lambda=1}^{q} tn_\lambda, \sum_{\lambda=1}^{q} fn_\lambda\right) \tag{3.5}$$

Tables 3.4 and 3.5 show the result of computing TP, TN, FP, and FN that was obtained from a classifier for the classes sport and economics. From Tables 3.4 and 3.5, the macro average precision is:

$$Macro\,Averaged\,Precision : (0.6 + 0.88)/2 = 0.74$$

Using the precision and recall from Table 3.6, the micro average is:

$$Micro\,Averaged\,Precision : 92/110 = 0.84$$

Micro averaging can be dominated by more common classes and would therefore work better for someone who is most interested in identifying as many classes as possible and who would be unconcerned that they might misidentify potential classes in smaller numbers. Other people, on the other hand, might be concerned with the less-common languages as the dominant and would likely employ a macro averaging-based evaluation.

3.3 Time Efficiency

Algorithms in computer science are analyzed by how much resources are required for the algorithm to finish the required task. These resources can be anything like time and space [Ausiello et al., 2012]. Run time analysis is one of the most popular algorithm analysis methods and implies how time grows with the input size. Usually it is measure in real time units but also using the main operation and counting them. Obviously, the correct choice of the algorithm for the problem can save a lot of time.

Data size (KB)	Algorithm A (in seconds)	Algorithm B (in seconds)
5	70	5
10	100	10
50	170	50
100	200	100

Table 3.7: Example running time of two algorithms A and B.

Data size (KB)	Algorithm A (in seconds)	Algorithm B (in seconds)
5	70	5
10	100	10
50	170	50
100	200	100
250	240	250
500	270	500
...	...	...
100,000	500	10,000
1,000,000	600	1,000,000
10,000,000	700	10,000,000
...	...	...
$63,072 * 10^{12}$	750	$31,536 * 10^{12}$ or one year

Table 3.8: Example running time for algorithms A and B.

For example, in Table 3.7, we are comparing two algorithms A and B. By just looking at Table 3.7 we can conclude that Algorithm B is much faster than Algorithm A.

By increasing the size of data and watching the running time, we understand that our conclusion was wrong. Algorithm B is more efficient than algorithm A on small input sizes (10 KB). With the increase of input data to a much bigger and sufficient large number, algorithm A shows its superiority over algorithm B. Table 3.8 shows the running time both algorithms for larger data sizes.

Obviously, the reason for change in behavior is that algorithm B running time increases linearly with the input data size. So, if we double the input data size, the running time doubles. For algorithm A, running time has logarithmic increase rate. This means that by increasing the input data size four times, the running time will only increase by a small constant number not by four times as in algorithm B. From that we conclude that, although algorithm B is better on small data, algorithm A will eventually show its superiority on input data sizes that are practical and for real world problems.

Nevertheless, comparing just running time might not be enough as the result of both algorithms might have different quality. So there are several trade-offs that may appear when we trade quality for speed and vice-versa. We explore this paradigm in the next section.

3.4 Trade-off Analysis

Can we trade quality and time and at the end improve both quality and time? Many times the answer is yes. Let us consider the following example that comes from NER. Let us say that algorithm A finds αn true entities in a text of size n in linear time while algorithm B finds $(\alpha + \varepsilon)n$ true entities in $(n \log(n))$ time. That is, B has better quality by a margin of ε, but

Data size (KB) n	**Algorithm A** (in seconds)	**Correct Entities Algorithm A**	**Algorithm B** (in seconds)	**Correct Entities Algorithm B**
n	$O(n)$	αn	$O(n \log n)$	$(\alpha + \varepsilon)n$
5	5	4	3.45	4
10	10	8	10	9
50	50	40	84.94	43
100	100	80	200	85
250	250	200	599.48	213
500	500	400	1,349.48	425
1,000	1,000	800	3,000	850
2,000	2,000	1,600	6,602.05	1,700

Table 3.9: Example running time for Algorithms A and B on the same data size.

Time	**Data size** (KB) n	**Algorithm A** (in seconds) $O(n)$	**Correct Entities Algorithm A**	**Algorithm B** (in seconds) $O(n \log n)$	**Correct Entities Algorithm B**
10s	-	10	8	10	9
85s	-	85	68	85	43
200s	-	200	160	200	85
3000s	-	3,000	2,400	3,000	850

Table 3.10: Example of correct entities of Algorithms A and B on the same running time.

is slower than the A. Therefore, the two algorithms running time might take something like the example shown in Table 3.9, where we consider $\alpha = 0.8$, $\varepsilon = 0.05$. By considering this data, we can conclude that algorithm B is better than algorithm A because finds more correct entities. However, we are just looking at quality with respect to the data size.

To compare them fairly, we need to consider the same time. So if both algorithms run time proportional to T, we have:

$$\begin{aligned} &\text{Algorithm A:} \quad O(\alpha n) \in O(T) \\ &\text{Algorithm B:} \quad O(\alpha + \varepsilon)\frac{n}{\log_2(n)} \in O(T) \end{aligned}$$

Hence, we can equate the two cases to find n such that:

$$\begin{aligned} \alpha n &> \frac{K(\alpha + \varepsilon)n}{\log_2(n)} \quad \text{for some constant } K \\ \log_2(n) &> K\left(1 + \frac{\varepsilon}{\alpha}\right) \\ n &> 2^{K\left(1+\frac{\varepsilon}{\alpha}\right)} \end{aligned}$$

That is, for a large enough n, algorithm A finds more correct entities than algorithm B, just because it can process more data in the same time, despite having less quality [Baeza-Yates, 2013]. Using now time as the parameter there is a value of n where algorithm A can find more correct entities than algorithm B as shown in Table 3.10, that considers $K = 100$.

Following this example, we now present the different analyses that we should perform to compare a set of algorithms on text processing problems. These analyses can be also used for many other problems.

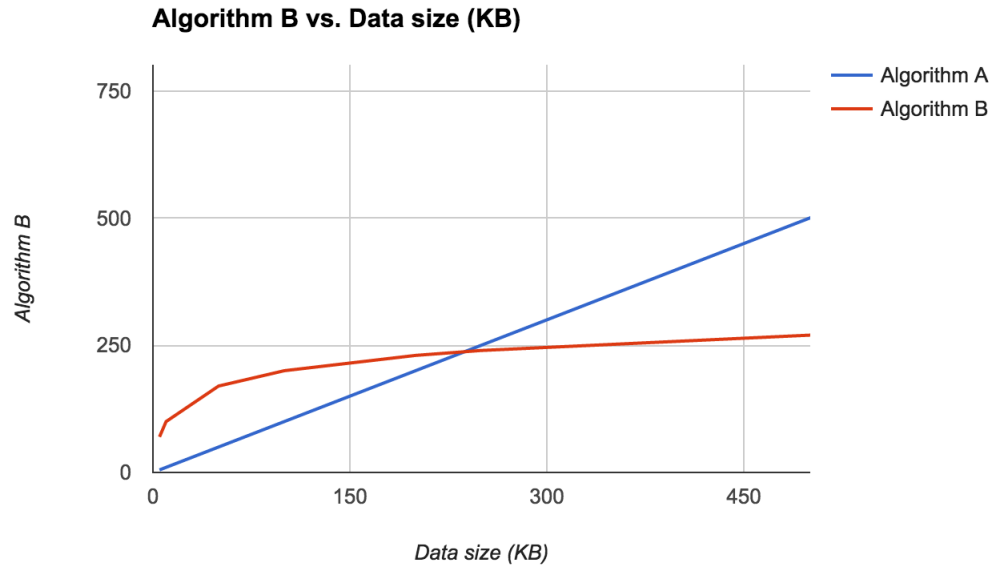

Figure 3.1: Running time vs. data size.

3.4.1 Quality and Data Size

We divide a dataset into different sizes of data. We run various models on each data size. Next, we calculate the quality of models by computing their F-measures. Then we generate a curve for each model that shows classification accuracy as a function of data size. By evaluation of the trade-off between quality and data size, we can find a point on the curve where quality will no longer get better with the increase of data size. This shows the nature of errors that exist at this point. Figure 2.10 is an example of trade-off between F-measure and data size for confusable disambiguation.

3.4.2 Time and Data size

With increasing data size, running time will increase. Increasing running time is dependent on growth rate of the order of a mathematical function. Each method has a different growth rate. We will review and compare growth rates of different methods. Figure 3.1 is an example of the growth rate obtained from Table 3.9.

3.4.3 Quality, Time and Data Size

As an example of a comparison in both quality and time, we measure the quality of different classification methods on various data sizes as well as the running time for classification methods on a MacBook Pro laptop with processor Core i7 2.5GHz Intel, with 1600 MHz DDR3 cache CPU and 16 GB of RAM memory. Table 2.5 presents the accuracy and running time on Basic Stanford Dependencies. In the last column we use the ratio $\frac{Quality \times Tokens}{Time}$ to evaluate these two factors together as shown in Table 3.11.

By only considering accuracy, Charniak-Johnson has better quality as shown in Figure 3.2. By considering just running time, Huang is faster. Figure 3.3 shows the result of quality and running time, where we show a practical case of a lesser quality algorithm is better than the best quality algorithm just because is much faster. Indeed, considering this ratio Charniak-Johnson is the before last performing algorithm.

Stanford dependencies	Accuracy	Tokens per second	Quality per tokens/second second
Stanford EnglishPCFG	90.06	123.63	11,134
Huang	90.90	**616.55**	**56,044**
MSTParser	91.24	239.60	21,861
Full TurboParser	92.29	209.98	19,379
Stanford RNN	93.11	66.57	6,198
Berkeley	93.33	200.00	18,666
Charniak-Johnson	**93.91**	100.72	9,458

Table 3.11: Accuracy vs. time on computing Stanford dependencies.

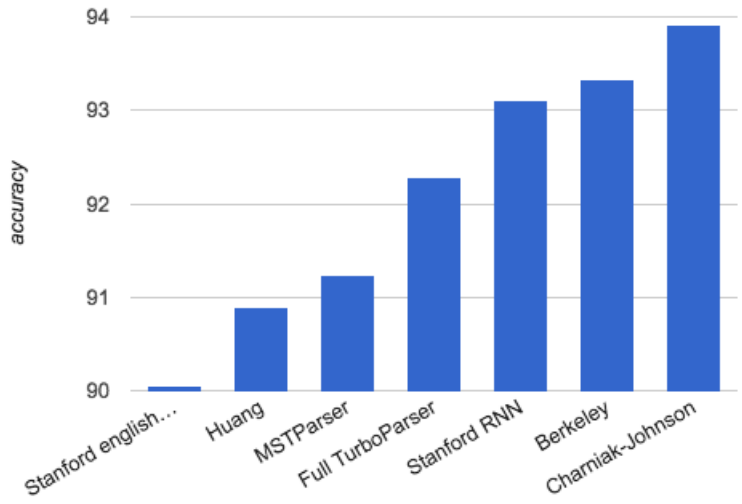

Figure 3.2: Accuracy on basic dependencies.

Based on this example we define our *performance* measure as:

$$Performance = \frac{Quality * Size}{Time} \tag{3.6}$$

which combines quality, time, and data size. That is, performance scales with the size of the data and penalized by the time consumed. This way, high quality on large datasets and low time will have very high performance, but high quality on large datasets with high time will decrease the performance and the same for low quality on large datasets with low time. Figure 3.4 is an example of performance change for five well-known classification algorithms applied to sets of data of different size. Here you can see that slower algorithms like KNN and DT decrease their performance while linear algorithms keep the performance more stable.

3.4.4 Dominant Algorithms

An important concept related to trade-offs is a *dominant algorithm*. A trade-off graph is a powerful tool for making decisions and usually when one measure improves, another decrease. Using the example given in Figure 3.5, we can show the dominant algorithms by drawing the frontier for them as in Figure 3.5.

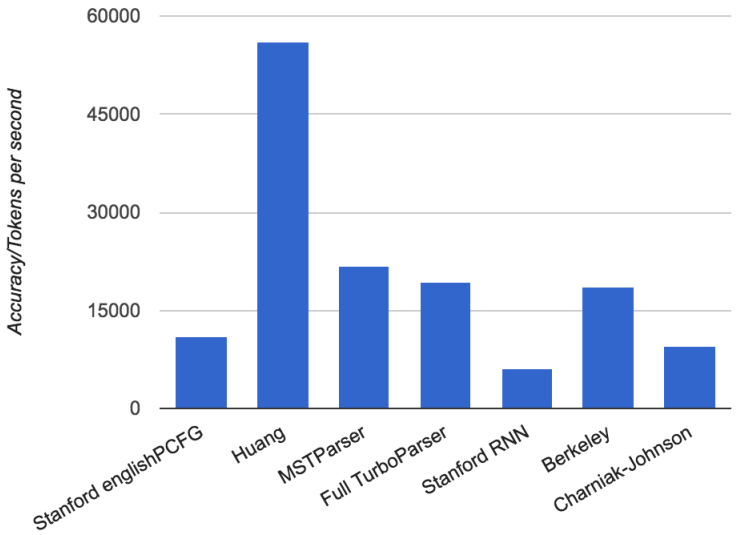

Figure 3.3: Accuracy vs. tokens per second.

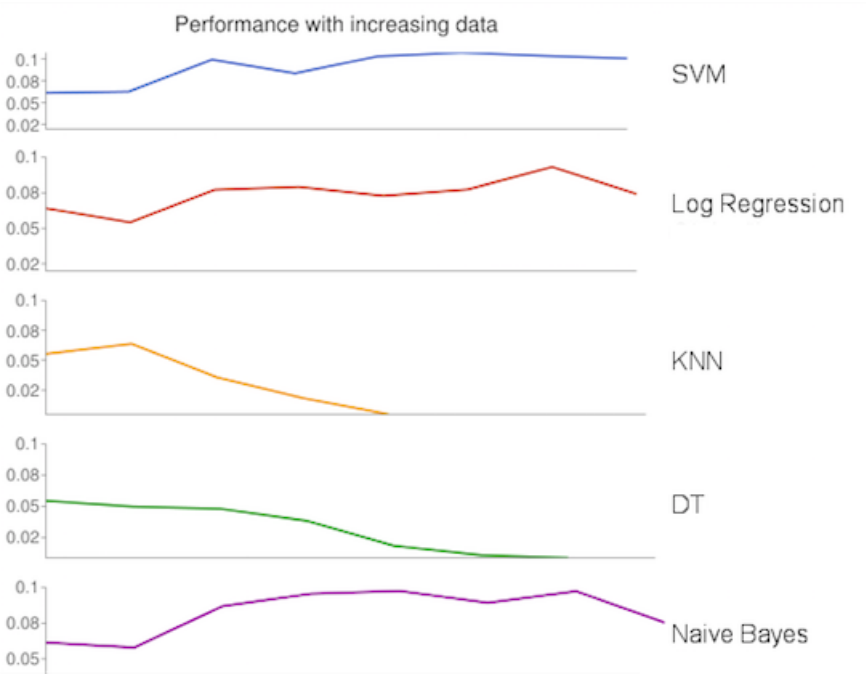

Figure 3.4: Performance vs. data size.

At one extreme, a choice like Huang would be selecting high tokens per second, but smaller accuracy. At the other extreme, a choice like Charniak-Johnson would be selecting a high level of accuracy, but less tokens per second. According to the graph, an increase of quality involves most of the time a loss in speed. However, we need to avoid choices like Stanford EnglishPCFG or Stanford RNN, that are not dominant in any measure. Efficiency requires that the choice is in the frontier where the dominant algorithms are.

Dominant algorithms usually are the same for all data sets, that is, the dependency on data is low. However, this is not always the case. When we include the number of data sets for which each algorithm is better, the notion of dominant algorithm gets more complicated. We

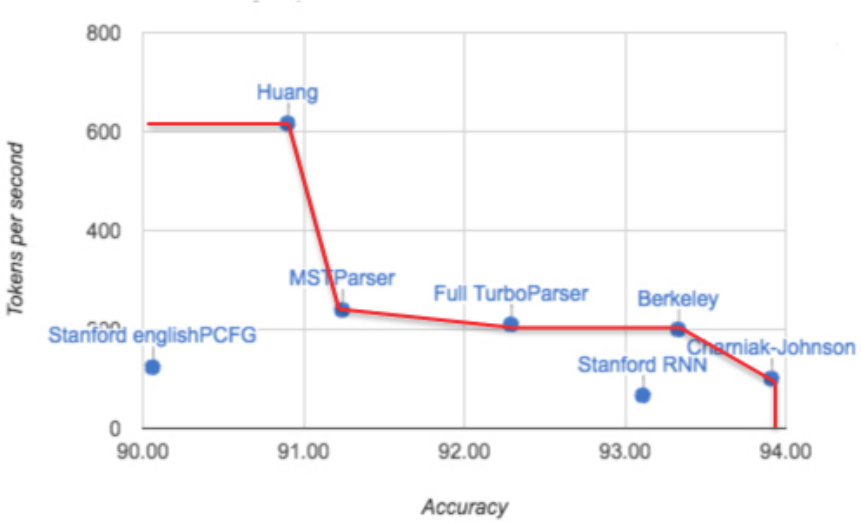

Figure 3.5: Example for dominant algorithms.

do not use this more complex notion, but a good example can be found in [Tax et al., 2015].

It is hard to find an algorithm that improves in both measures (in this case, tokens per second and accuracy quality) together. We usually must make difficult choices between two dominant algorithms, where one is dominant in tokens per second and the other is dominant in quality. However, we should prefer to choose an algorithm with high efficiency. That is, the choice is on the frontier curve rather than inside it. To be able to compare dominant algorithms we need to define a new measure that relates time and quality. The best algorithm would be the one that dominates all other algorithms in all the measures considered. However, this seldom exists as it is hard to improve upon all algorithms on all measures.

3.5 Methodology

Our methodology follows the model that is shown in Figure 1.2. However, we had to adapt this model to our framework. Our model consists of the steps described next.

3.5.1 Data Collection

This step selects subsets of data from all the available data that we will be working with. We choose increasing size subsets until reaching the complete dataset. For example, 10, 50, 100, 500, 1000 MB.

3.5.2 Defining the Target Variable

We define a target variable depending on the problem to be solved and documents to be categorized. We define a target variable that answers a question such as:

- In NER: “Which entity class does a word belong to?”
- In news classification: “Which topic does a document belongs to?”

- In sentiment analysis: “Does a product have a positive review or a negative one (0/1)?”
- In patent classification: “Is a patent to be accepted or rejected (0/1)?”

Notice that we have chosen two binary class problems and two multi-class problems.

3.5.3 Pre-process Data

After we have selected the data and defining a target variable, a step to change this data in a usable form is needed. Pre-processing then implies putting the data in a format that is more user friendly in our system. We can divide this step into the following sub-steps:

3.5.3.1 Document Representation

One of the pre-processing phases is to change the document representation to reduce complexity and the high dimensionality in text features from the documents [Khan et al., 2010]. It makes it easier to handle by transforming a full text format to a vector.

3.5.3.2 Target Preparation

Before using the dataset in classification, we must prepare the target class first.

3.5.3.3 Data Cleaning

Real world data tends to be noisy, incomplete, and inconsistent. In this step, we remove abnormalities in the data in the form of: noise, outliers, missing values, extreme cases, and inconsistencies.

3.5.3.4 Feature Extraction and Selection

First, we extract what can be used as features for the following steps. After feature extraction, the next important technique we must perform Feature Selection (FS). Since the text features are high dimensional, Feature selection is used to improve efficiency and scalability of the classifier by creating, for example, a vector space model.

3.5.4 Classification Methods

Depending on the problem, we use and implement different classification methods. The classification system can be divided into two sub processes: Learning and Testing. There are several ways to implement those two steps with Cross-validation and Hold-out test being two of these ways. The data size and the type of problem to solve will dictate which one of those methods to use.

3.5.5 Evaluation Metrics and Validation

There are several evaluation methods for classification such as confusion or error matrix sensitivity, specificity, and overall accuracy. Common performance measure just counts errors by using Precision, Recall, and F-measure. Running time and data size also need to be considered. We use F-measure, precision, and recall for our evaluation of quality. For the trade-off analysis, we consider running time and size of data as mentioned earlier.

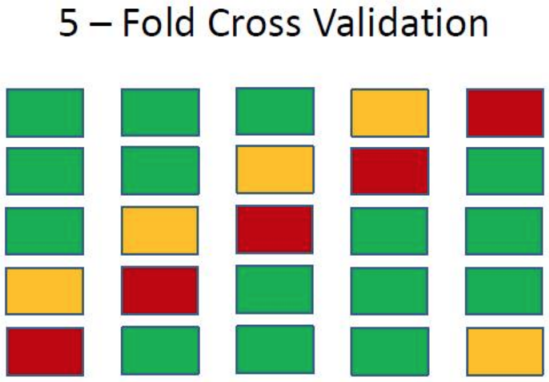

Figure 3.6: Example of a 5-fold cross validation scheme.

We use two evaluation techniques. A simple one, holdout, that divides the data randomly in two sets. One for training with 80% the data and one for testing with 20% of the data. For data sets with a small number of examples and less attributes, an arbitrary split would result in both small training and test sets, potentially yielding varied results for different ways of splitting. Paired t-test is used to measure the significance of accuracy differences [Yu, 2008].

We also use the K-fold technique with exhaustive cross-validation, which takes more time but it is more robust. Cross-validation splits a data set into K folds and runs the experiment K times. Each iteration, $K-1$ fold of the data is used for the training process, while 1 fold is used for the testing process. The accuracy is averaged over the results of the K runs. Two advantages of this technique are:

1. The model will be less prone to overfitting.

2. The model can generalize better to unseen data.

To use K-fold we first need to split the data in K random subsets of size as equal as possible. Then we iterate over the K subsets from $i = 1$ to K such that:

1. Dataset partitioning:

   - Testing dataset is subset i.
   - Training dataset is the union of the other $(K-1)$ subsets.

2. The "Training Dataset" is used to prepare the model

3. The "Testing Dataset" is used to evaluate the model, hoping that the same performance will be achieved in unseen data.

3.5.6 Trade-off Analysis

After running different ML algorithms on various data size, we analyze and compare the results using the framework defined in Section 3.4. The final methodology is shown in Figure 3.7.

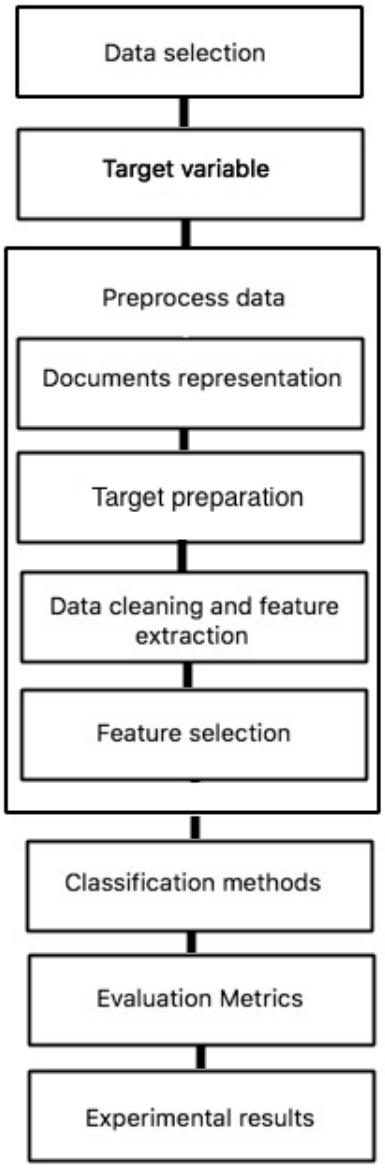

Figure 3.7: Knowledge discovery process.

3.6 Discussion

In this chapter, we have presented our trade-off analysis framework. We talked about trade-offs between quality, time, and data size. We discuss how these trade-offs can affect when choosing algorithm. We explained the notion of dominant algorithm and how we can use it to find effective algorithm. We showed the steps of our analysis methodology. Now we explain the experimental rationale that we use in the following three chapters for each of the problems that we selected. As we mentioned before, the three problems have different granularity, from words to full documents. The datasets used also have different sizes due to the availability of large public datasets for each problem. We also use two different evaluation techniques, the simple holdout one or K-folds. Finally, in one case we consider two subsets of the same dataset (reviews) and two classes; while in the other case we consider two completely different collections (news and patents) and different prediction problems (multi-class or binary). As we cannot exhaustively compare all the parameters, this selection has good coverage of all possible

Dataset	Objects	Size	# Features	Evaluation
News	Documents & words	2.13 GB	11	6-Folds
Movies & TV	Reviews	8.77 GB	10	Training/test
Books	Reviews	14.39 GB	10	Training/test & 10-Folds
Patents	Documents	7.18 GB	6	Training/test & 6-Folds

Table 3.12: Raw datasets, features, and evaluation techniques used for our experiments.

Problem (Object)	Classes	Dataset	Algorithms
NER (words)	Multi	News	SNER, LingPipe, Illinois
Sentiment Analysis (reviews)	Binary	Movies & TV	DT, LR, KNN, RF, SVM
	Binary	Books	DT, LR, KNN, RF, SVM
Document Classification (documents)	Multi	News	DT, LR, KNN, Naïve Bayes, SVM
	Binary	Patents	DT, LR, KNN, RF, SVM

Table 3.13: Problem, datasets, and algorithms considered.

combinations. For each problem, we choose several algorithms, where we tried to have two problems with a similar set algorithms to see the effect of the problem on the results. Another parameter of the experimental space is the number of features, but we decided to keep that one of the same order of magnitude for all cases as this parameter has a much larger granularity than the other parameters.

We summarize these experiments using Tables 3.12 and 3.13. In Table 3.12 we show that the rationale is that we start classifying words, then small documents (reviews) and then documents. For this reason, we use a news dataset for word classification, the Amazon reviews dataset (Movies & TV and Book) for sentiment classification (hence two homogenous subsets for a binary prediction problem), and patents and news for document classification (the first binary and the second multi-class). We also give the maximal size of the datasets and the evaluation techniques used.

In the next three chapters, we present the results for the experiments in the three problems that we have selected: Named Entity Recognition, Sentiment analysis, and Document Classification. The characteristics of each problem, datasets and algorithms used for each one are shown in Table 3.13. For all the experimental results, we used a MacBook Pro laptop with processor Core i7 2.5GHz Intel with 1600 MHz DDR3 cache CPU and 16 GB of RAM memory.

Chapter 4

NAMED ENTITY RECOGNITION

4.1 Introduction

This chapter introduces the Named Entity Recognition (NER) problem for text, presents the existing available named entity extractors, and describes the experimental setup and kind of data that was used for this work. At the end, we present the experiments to compare existing NER classification methods on different data sizes and the results that were obtained.

There are many proposed technologies for language problems such as Summarization, Parsing, Part-of-Speech-Tagging, Named-Entity-Recognition, Word-Sense Disambiguation, and Machine-Translation [Dudhabaware and Madankar, 2014]. Recognizing and extracting such data is a core process of natural language processing. Named Entity Recognition is a popular domain of natural language processing [Urbansky et al., 2011]. We select Named Entity Recognition technology because it is one of the main technologies used for prepossessing step on more advanced technology in NLP and it works on different kinds of corpus. Some NLP problems that use NER are shown in Figure 4.1.

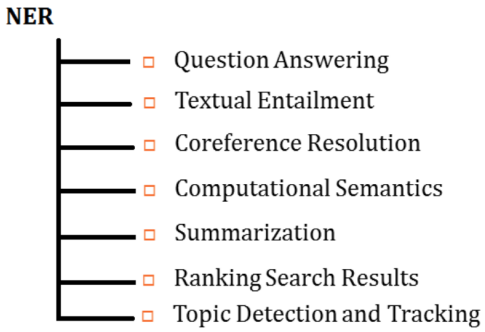

Figure 4.1: Problems that need NER.

NER is a process that is defined as categorizing and identifying strings of text into different classes [Urbansky et al., 2011]. Examples of such classes are:

- Persons,
- Locations,
- Organizations,

- Objects,
- Date and Time,
- Money,
- Percent,
- Currencies,
- etc.

NER task labels sequences of words in a text for different languages such as English, Spanish, French, Portuguese, Chinese and Japanese. For this reason, many systems exist and each designed to perform this type of task differently. Additionally, they also differ in processing methods [Finkel et al., 2005]. NER systems differ greatly and the task of selecting one for a specific situation can be difficult due to the variance of the entity types the system can detect, the nature of the text, and the nature of its input and output.

The next section briefly summarizes the existing available NE systems and then we try to compare time complexity and performance of these systems in different size of data.

4.2 Algorithms

As mentioned before, it is our intent to evaluate the performance (quality and time) of existing NER systems by different sizes of data. Many methods and systems were designed for NER such as Factorie, Illinois NER, Vineet Yadav, Stanford NER, GATE ANNIE, Minor Third, OpenCalais, Lingpipe, Mallet, Alchemy, and Opener. There are several commercial systems with good performance, but we focused on those with license for research and public. In general, NER methods divided into three main families [Mansouri et al., 2008]:

- **Hand-made rule-based methods (HRM):** HRM use manually constructed finite state patterns.
- **Machine learning-based methods:** Machine learning treat NER as a classification process.
- **Hybrid methods:** The hybrid methods use a mix of those two approaches.

Many machine learning techniques have been applied including maximum entropy models, hidden Markov models, decision trees, conditional random fields, and support vector machines. Also, different researchers have shown that combining several weak classifiers produce a stronger single classifier [Urbansky et al., 2011]. Machine learning methods contain three approaches used to recognize previously unknown entities:

- **Supervised:** The current powerful technique for addressing the NERC problem is Supervised Learning (SL). Supervised learning relies on reading and training large annotated corpus, memorizes lists of entities, and creating disambiguation rules to recognize previously unknown data [Nadeau and Sekine, 2007]. There are multiple techniques in supervised learning for instance: Support Vector Machines (SVM) [Asahara and Matsumoto, 2003], Decision Trees [Sekine, 1998], Hidden Markov Models (HMM) [Bikel et al., 1997], Conditional Random Fields (CRF) [McCallum and Li, 2003b], and Maximum Entropy Models (ME) [Borthwick et al., 1998].

System	Language	Interface	License	Entity Types
SNER	Java	Console, Java	GPI. V2	People, Location, Organization, Money, Percent, Date, Time
INET	Java	Console, Java	Research and Academic use license	People, Locations, Organization, plus 15 other types based on Ontonotes
LIPI	Java	Console, Java	Free and commercial Licenses	People, Locations, Organizations

Table 4.1: Summary of NER systems.

- **Unsupervised:** Unsupervised learning relies on statistics and lexical patterns to identify certain rules on large unknown data. One very popular example of unsupervised learning is clustering where data must be clustered into groups based on common similarity features. To be able to train an unsupervised learning, a learned unannotated corpus is needed [Nadeau and Sekine, 2007].

- **Semi-supervised:** Semi-supervised learning or Weakly supervised is the best of two worlds. It relies on a large unannotated corpus with a small annotated part called a set of seeds. The main idea behind semi-supervised learning is to use the set of seeds to build the basic knowledge then search the unannotated part of the corpus to find the items matching the seed or having similar features and context to the seed. When the system finds similar features, it uses this information to expand its knowledge and the learning process is repeated until no new knowledge is added. At the end of the learning process, many items matching the seed would be identified [Nadeau and Sekine, 2007].

We selected three supervised NER systems that are publicly available, well known, free for research, and are based on Machine learning methods for comparison:

- Stanford Named Entity Recognizer (SNER)

- Illinois Named Entity Tagger (INET)

- LingPipe (LIPI)

The summary comparison of the systems is shown in Table 4.1.

4.2.1 Stanford NER

Stanford NER (SNER) [Finkel et al., 2005] is a Java implementation of a Named Entity Recognizer. This popular Java-based system is based on linear chain Conditional Random Fields which is a supervised learning method. It provides several predefined models for the English language. Even if these models are not useful for our purposes, we can use dictionaries during the training phase.

The first model SNER1 is based on the CoNNL03 training set, and can recognize person, location and organization entities, and a generic type called Misc [Finkel et al., 2005]. The second, SNER2 is prepared and trained with the MUC6 and MUC7 corpora, and can handle seven entity types: Time, Location, Organization, Person, Money, Percent and Data. The third, SNER3 is trained on all corpora plus ACE, and is able to recognize person, location and organization entities. Each of these three models can be found in a plain and augmented version which includes distributional similarity features. Therefore, we used only SNER3.

4.2.2 Illinois Named Entity Tagger

This Java-based system [Ratinov and Roth, 2009] is based on three supervised learning methods: multilayered neural networks, hidden Markov models, and other statistical methods [Ratinov and Roth, 2009]. It also uses manually annotated dictionaries for lookup and word clusters generated from unlabeled text to improve performance. A few word clusters and dictionaries are distributed with the system, and it is possible to build new ones, such as word clusters, models, and output encoding. This system has been trained in several models to provide English texts from the CoNLL03 corpus. Thus, they can detect person, organization, location, and miscellaneous. entities. INET allows for training new ones. The first model (INET1) was generated to have a lower bound when compared to the performances of the other configurations. The second (INET2) is the result of a single-pass process. The third (INET3) was obtained through a two-pass process; it is supposed to be better, but remains slower. The fourth model (INET4) is based on the same process, but it was trained on both CoNLL03 training and development sets. By comparison, the three other models relied only on the training set. The Illinois is based on conditional random fields [Atdağ and Labatut, 2013].

4.2.3 LingPipe

This system -Alias-i2008- is commercial and can handle various other NLP tasks besides NER. It has free licensing available for academic use and it is open source. It relies on n-gram character language models, trained through hidden Markov models [Mansouri et al., 2008, Labatut, 2013]. Three different models are provided for the English language. Two of them are dedicated to genetics-related texts, which have very little interest to us. The third is built on the MUC6, CoNLL03 corpus and can detect Organizations, Locations and person's entities. Many aspects of the process, such as the chunking method, can be controlled via a configuration file. Named entity recognition model in LingPipe involves multiple methods that are designed to work together smoothly. These methods can be a supervised training or regular expression matching or even dictionary matching. [Atdağ and Labatut, 2013]

In the next section, first we explain available datasets for NER. Then we explain a selected dataset for our experiment. The experimental setup is presented in section 4.3.

4.3 Dataset: News

NER requires a big amount of data for training and testing the systems. Most research uses some standard corpora which was designed for conferences or competitions. These corpora focus on certain types of text, such as news, military, emails, and terrorism. Table 4.2 shows the comparison of the available corpora for NER and their properties such as the language that corpora support, Number of Entity types, domain and availability of corpora.

Corpus	Size in words	Language	Entity Types	Domain	Access
NYTAC	N/A	English	4	News	Commercial
MUC1	N/A	N/A	N/A	Military messages	Unavailable
MUC2	N/A	Chinese, Japanese	10	Military messages	Public
MUC3	N/A	English	18	Terrorism reports	Public
MUC4	N/A	English	24	Terrorism reports	Public
MUC5	N/A	English, Japanese	47	International trade	Unavailable
MUC6	N/A	English	6	Negotiators, management	Commercial
MUC7	N/A	English	7	Aeronautics, weaponry	Commercial
NIST IE-ER 99	N/A	English	6	News	Unavailable
CoNLL02	N/A	Dutch, Spanish	4	News	Commercial
CoNLL03	N/A	English, German	4	News	Commercial
Email Corpora	N/A	English	1	Emails	Unavailable
ACE1	225 k	English	5	News	Commercial
ACE2	270 k	English	5	News	Commercial
ACE2003	150 k	Arabic, Chinese, English	5	News	Commercial
ACE2004	200 k	Arabic, Chinese, English	7	News	Commercial
ACE2005	310 k	Arabic, Chinese, English	7	News	Commercial

Table 4.2: Comparison of the corpora properties [Atdağ and Labatut, 2013].

Most editions of the conference on Computational NL Learning (CoNLL) host a NLP related competition, and provide data sets to evaluate the proposed systems. In 2002 and 2003, this shared task was NER. Both corresponding corpora are composed of news texts, which are annotated using the entity types: Organization, Person, Location and Misc. Texts are divided into three groups: a training set and two test sets. The first test set is used for training the Machine Learning algorithm, whereas the second one is reserved for the final evaluation of the system and is supposed to be more difficult to process. CoNLL02 only contains Dutch and Spanish texts, but CoNLL03 focused on the German and English languages as shown in Table 4.2 on the line labeled [Erik and Fien, 2003].

The annotations are publicly available, but their use requires access to commercial corpora [Lewis et al., 2004]. Figure 4.2 shows part of CoNLL03 dataset, directories and XML files.

This data set is a collection of newswire articles from the Reuters Corpus [Lewis et al., 2015] and is distributed via web download containing about $810,000$ XML Reuters in the

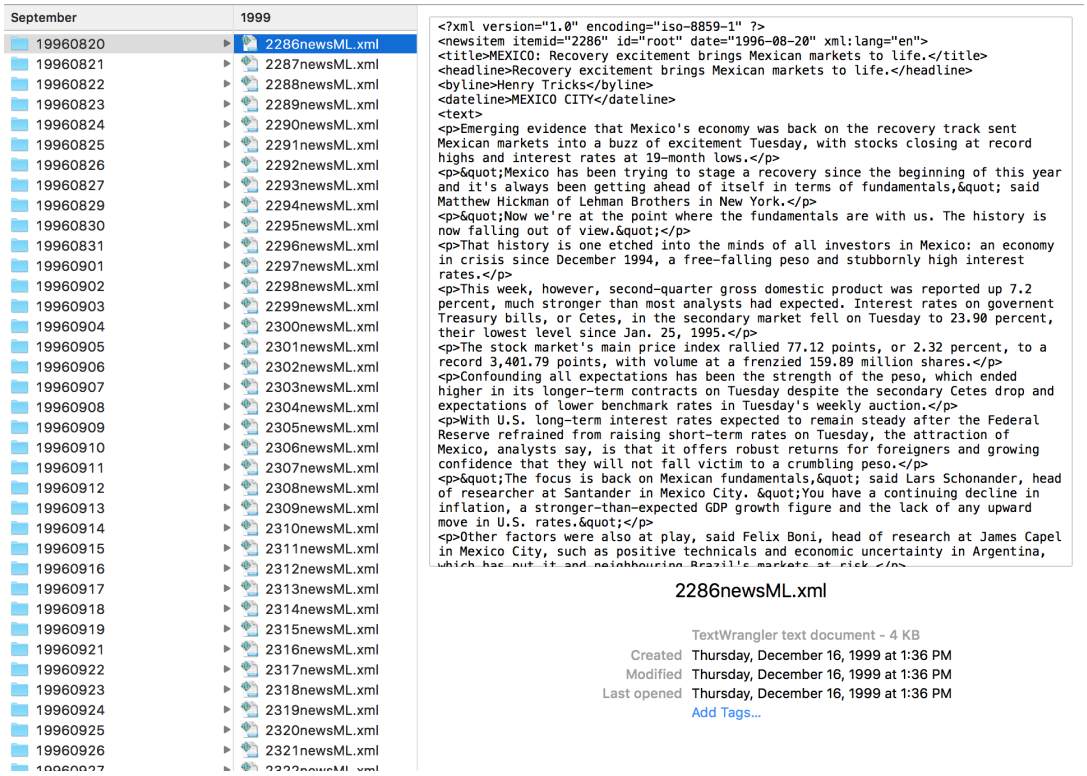

Figure 4.2: Example of the CoNLL03 dataset.

```
Mexican NN I-NP L
markets TO I-VP O
to VB I-VP O
Henry NNP I-NP I-PER
Tricks NNP I-NP I-PER
MEXICO NNP I-NP I-LOC
CITY NNP I-NP I-LOC
```

```
U.N.          NNP  I-NP  I-ORG
official      NN   I-NP  O
Ekeus         NNP  I-NP  I-PER
heads         VBZ  I-VP  O
for           IN   I-PP  O
Baghdad       NNP  I-NP  I-LOC
.             .    O     O
```

Figure 4.3: Sample of tokenized output generated by a system.

English Language. It takes approximately 2.5 GB of uncompressed file storage space. The data files start with a line containing the keyword -DOCSTART- to identify the beginning of each article. Empty lines are used to make sentence boundaries. The data files contain four columns. The first column is used for the word. The second column is used for Part-Of-Speech (POS) tag. The third column is used for syntactic chunk tag which can be either I-TYPE (Inside a phrase), B-TYPE (Beginning of a phrase), O-TYPE (Other). The fourth column is used for a named entity tag.

Each non-empty line contains the following tokens shown in Figure 4.3:

- Current word
- Lemma of the word (German only)
- Part-Of-Speech (POS) tag generated by a text tokenizer
- Named entity tag given by human annotators

4.4 Experimental Setup

First and foremost, NER needs to build a knowledge base which will contain the known named entities. Then it tries to link an entity to a knowledge base entity node or no match. This process can be separated into three main components: extractors, searchers and disambiguation.

4.4.1 Extractor

In the extraction phase, different techniques are used to detect the named entities. These techniques include tokenization, part of speech tagging, finding sentence boundaries, and in-document co-reference. From these techniques, some can be general such as tokenization and capitalization, However, some of these can be very problem specific like in-document co-reference where the extractor is trying to detect named entities that point at the same entity. For example, NBA stands for National Basketball Association.

4.4.2 Searcher

Searcher is the process of generating a set of candidates for a named entity to be linked to. This is done by querying an index over these aliases (potential mention strings that can refer to an entity for each article). This index can be disambiguation pages, titles, and other Wikipedia-derived aliases to capture synonyms. A good searcher should create a small set of candidates for each mention string. This is done by balancing precision and recall. Maintaining a small set of candidates is beneficial later in the disambiguation phase by reducing the computation required.

4.4.3 Disambiguator

After the searcher has created a set of candidates for each mention, the disambiguator chooses the best entity for each mention bearing in mind that the best candidate is not always the first one returned by the searcher.

The entity types we chose to focus on are PERson (PER), LOCation (LOC) and ORGanization (ORG). First all the XML tags were removed from XML files and then we made a new text file. We divided data into different sizes (e.g. 1 MB, 5 MB, 10 MB, 50 MB, 100 MB, and 500 MB, 1 GB).

The NER systems that we selected all have different heap sizes and different types of input data. We increased the heap size and changed the source codes so we could upload the same data set for all systems. We ran the selected systems on different data sizes. Every system has a different output. We updated the source codes to have the same output. Then we cleaned the results obtained from the algorithms and extracted the information we needed as shown in Figure 4.4.

In NER, to evaluate how good the system is, human linguists are asked to identify the named entities and then compare their output with the model output to decide how good the system is. For example, Figure 4.5 shows an annotated text marked up as a solution to use to evaluate the model [Nadeau et al., 2007] and one of the systems producing this output is shown in Figure 4.6. The systems produce five different errors, explained in Table 4.3.

The scoring protocol is "Exact-match evaluation" where the system only scores a guess as correct if it's an exact match of the entity in the solution. To compare systems, we use Micro-Average F-measure (MAF). To be able to calculate MAF, we need to calculate precision and

```
MEXICO LOCATION
Henry PERSON
Tricks PERSON
MEXICO LOCATION
CITY LOCATION
Matthew PERSON
Hickman PERSON
```

Figure 4.4: Sample clean tokenized.

```
Unlike <ENAMEX TYPE="PERSON">Robert</ENAMEX>, <ENAMEX TYPE="PERSON">John
Briggs Jr</ENAMEX> contacted <ENAMEX TYPE="ORGANIZATION">Wonderful
Stockbrockers Inc</ENAMEX> in <ENAMEX TYPE="LOCATION">New York</ENAMEX>
and instructed them to sell all his shares in <ENAMEX
TYPE="ORGANIZATION">Acme</ENAMEX>.
```

Figure 4.5: Sample of output solution [Nadeau and Sekine, 2007].

```
<ENAMEX TYPE="LOCATION">Unlike</ENAMEX> Robert, <ENAMEX
TYPE="ORGANIZATION">John Briggs Jr</ENAMEX> contacted Wonderful <ENAMEX
TYPE="ORGANIZATION">Stockbrockers</ENAMEX> Inc <ENAMEX TYPE="PERSON">in
New York</ENAMEX> and instructed them to sell all his shares in <ENAMEX
TYPE="ORGANIZATION">Acme</ENAMEX>.
```

Figure 4.6: Sample of output from a system [Nadeau and Sekine, 2007].

Correct solution	**System output**	**Error**
Unlike	Unlike as Location	False Positive, the system predicted an entity but there is not one.
Robert as Person	-	False Negative, the system could not predict an entity.
Jone Briggs Jr as Person	Jone Briggs Jr as Organization	False Negative, the system predicted an entity, but generated the wrong type.
Wonderful stockbrokers inc as Organization	Stockbrokers as Organization	False Negative, the system predicted an entity, but could not get its boundaries correctly.
New York as Location	In New York as Person	False Negative, the system predicted both the type and boundaries of the entity wrong.

Table 4.3: Example errors that were found by a NER system.

recall. For precision, we get the percentage of correct guesses out of all guesses. For recall, we get the percentage of correct guesses out of actual entities in the solution [Nadeau and Sekine, 2007].

For the previous example, the solution contains five entities. The system guessed five entities in which only one is exact match (ACME as Organization). The precision is $1/5 = 20\%$ and coincidentally the recall is also $1/5 = 20\%$, so the MAF is also 20%.

Word	Prediction	Correct Solution	Evaluation
MEXICO	LOCATION	LOCATION	TP
Henry	PERSON	PERSON	TP
Federal	ORGANIZATION	ORGANIZATION	TP
James	ORGANIZATION	PERSON	FP
look	PERSON	-	FN

Table 4.4: NER error types.

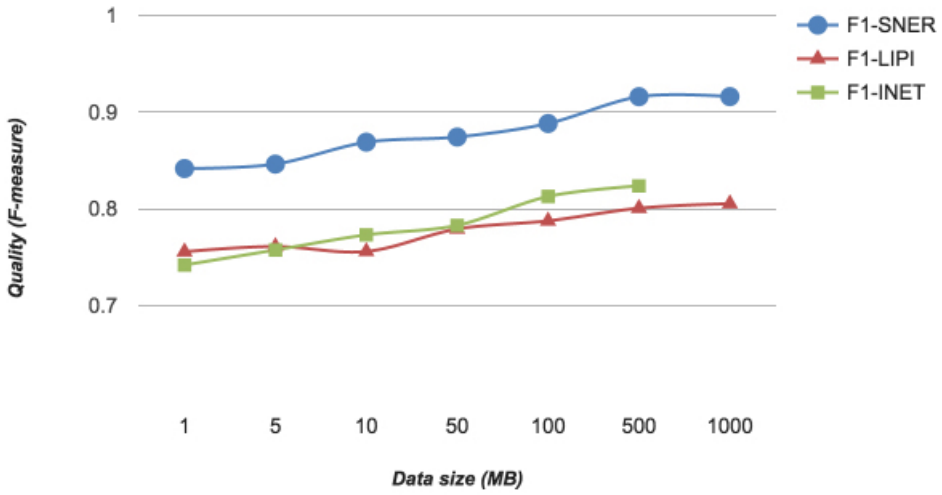

Figure 4.7: Comparison of recognizer based on different data sizes on CoNLL.

4.5 Experimental Results

There are various methods that determine effectiveness in classification problems, The common performance measure method just counts errors by using Precision, Recall, and F-measure. In this experiment, we used F-measure, Precision, and Recall. So, in the next subsections we show our results, following the proposed methodology.

4.5.1 Quality Comparison

To determine precision and recall, each term that the system classified must be rated as True Positive (TP) meaning it was correctly classified, False Positive (FP) meaning it incorrectly classified the term, or False Negative (FN) meaning it did not classify the term when it should have. Table 4.4 shows a sample of NER error types obtained from the CoNLL dataset.

Figure 4.7 shows the overall comparison of quality for the three NERs when used on different data sizes on the CoNLL data. Our result shows SNER had clearly better results than LIPI and INET in all data sizes. SNER has more capability to recognize and categorize the words correctly compared to LILP and INET. SNER has less FN and FP than other systems and that makes SNER perform better. In general, with increased data size, quality (F-measure) is increased. Once reaching 500 MB data size, the quality has stabilized and increasing the data did not improve the quality.

Size of data	NER systems		
	LIPI	SNER	INET
1 MB	0m38.6s	0.m22.8s	3m14.0s
10 MB	5m59.2s	1m21.01	32m27.8s
50 MB	28m54.3s	19m.36.5s	-
100 MB	-	48m.0s	-
$\geq$ 500 MB	-	-	-

Table 4.5: Scalability evaluation.

Data Sizes	NER systems		
	LIPI	SNER	INET
1 MB	0m38.2s	0m22.9s	3m6.7s
10 MB	5m53.7s	3m3.7s	21m33.8s
50 MB	22m17.8s	19m30.8s	81m43.3s
100 MB	59m10.4s	40m35.0s	145m33.8s
500 MB	291m16.2s	210m30.6s	831m48.2s
1 GB	575m57.4s	**402m9.8s**	-

Table 4.6: Improved version for handling larger data.

4.5.2 Time Efficiency Comparison

Table 4.5 shows the running time versus data size. The results show that the systems cannot run on bigger data sizes. By increasing the size of data, the time consumed (running time) is not efficient anymore. SNER has better speed than other systems on increasing size of data and can handle the size of data up to 100 MB. Once the size has reached 500 MB the SNER was killed. LIPI can work on up to 100 MB and after 21m18s of running time was killed. Illinois could run up to 10 MB, but after spending 74m40s in running time this system stopped on 50 MB data size.

We optimized the source code so the systems could handle larger data. Table 4.6 shows how the systems were now capable of running on larger data size. Unfortunately, we are still facing a large time complexity that make these systems inefficient. The results in Table 4.6 show that SNER has better speed than other systems and can handle a data size up to 1 GB. We can conclude that SNER can handle bigger data sizes than the other two systems. After SNER, LIPI is the second fastest and INET crashed after 500 MB data.

Figure 4.8 shows the running time of these systems in different size of data. SNER has the best running time compared with other systems. After 100 MB, running time increased by five times.

4.5.3 Dominant Algorithm

Figures 4.9 and 4.10 present a comparison between running time and accuracy in NER systems. In this experience SNER has better result in both axes. It is faster and has better quality. LingPipe and Illinois are not dominant in any measure. This comparison shows that SNER is more efficient than the any of the other systems in this experiment, being the unique dominant algorithm.

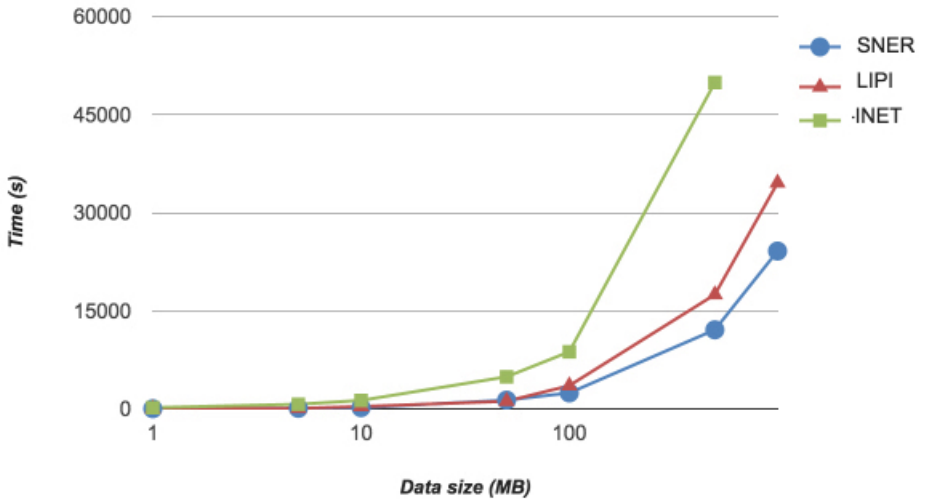

Figure 4.8: Running time vs. data size for all NER systems.

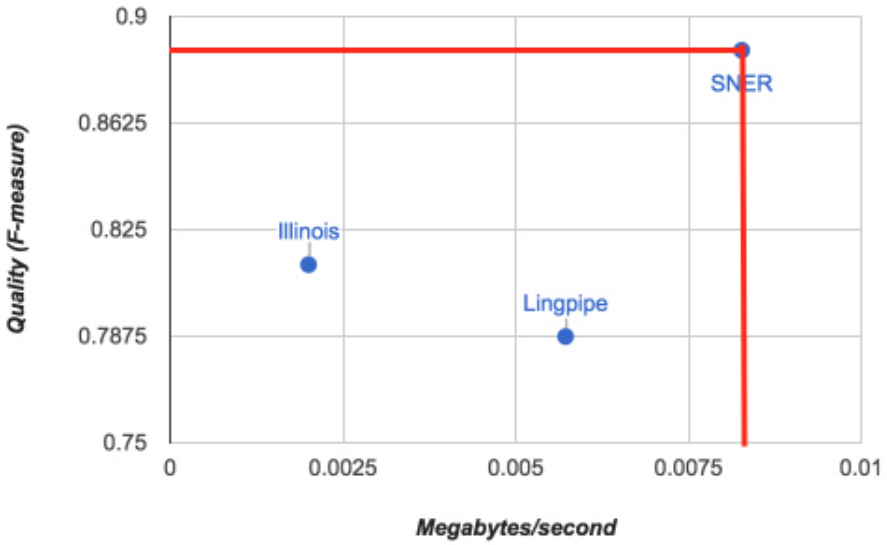

Figure 4.9: Dominant algorithm for NER systems on 100 MB.

4.5.4 Overall Comparison

Now we use our performance measure defined in Equation 3.4. Quality is scaled by size of data and compensated by the running time. The result of this equation is that a system that has high quality on large datasets while using less time is shown to be the more efficient system. However, high quality on a large dataset with more running time is less efficient, and low quality on a large dataset with less running time is also less efficient.

As you can see in the Figure 4.11, overall SNER has better performance, considering all three factors.

By drawing a chart for each size of data, we end up with a chart that shows the best system for each different data size. SNER has the best performance in all different sizes of data except for 50 MB. For 50 MB data, LIPI shows better performance, as this system is faster than SNER for that data size.

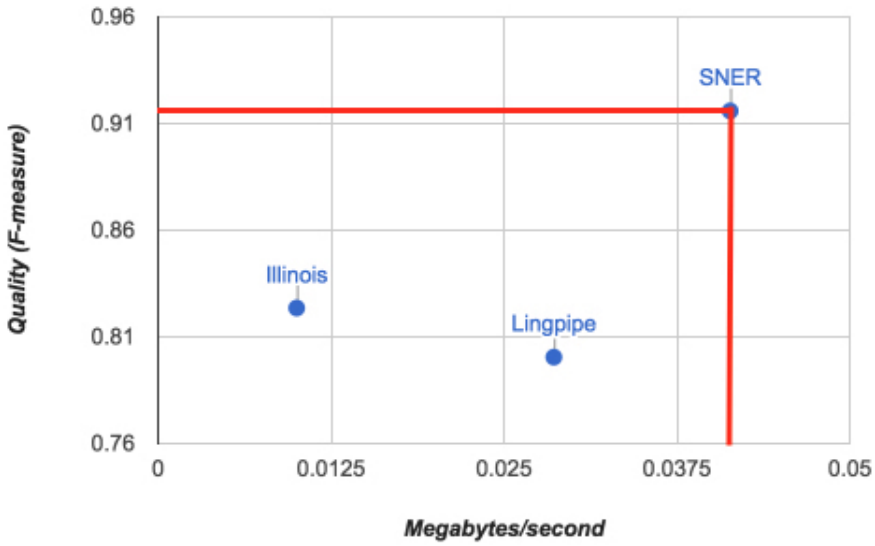

Figure 4.10: Dominant algorithm for NER systems on 500 MB.

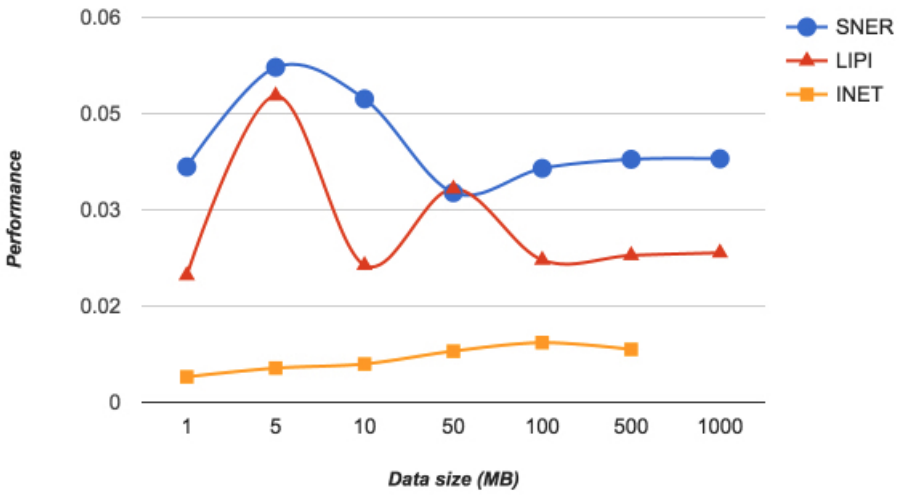

Figure 4.11: Performance and data size in NER.

4.6 Discussion

SNER shows the best quality and speed between all the NER systems that were compared in different data sizes. Hence SNER is the only dominant algorithm for all data sizes. Indeed, SNER shows the best quality and Megabyte/second between all the algorithms in different data sizes in the NER domain. Therefore, we can say clearly it is an efficient algorithm.

Considering the results obtained for 1 GB, SNER has the best performance, quality, and speed between the other systems. SNER is faster and shows better quality on CoNLL03 dataset. SNER took about 3.5 hours to run on 500 MB data and about 6.5 hours on the 1 GB dataset. INET crashed for 500 MB after several hours. LingPipe and INET have almost the same quality, but LingPipe is faster than INET in all data sizes.

Chapter 5

SENTIMENT ANALYSIS

5.1 Introduction

In the past few years, as the amount of customer reviews have grown rapidly on online product service websites, reviews earned their place as very effective for the decision making process. People tend to decide a purchase if the product is supported with good customer reviews. This fact encouraged a lot of research to extract the opinion from the online reviews. From here came the rise of sentiment analysis or opinion mining and it took its place to be one of the most important tasks of NLP gaining much focus from researchers and application developers [Fang and Zhan, 2015].

Sentiment analysis is used for identifying whether the query text is positive or negative and sometimes can be used to find the degree of positivity or negativity. The increase of online shops (Amazon, Ebay, Google Play, etc.) resulted in having a massive amount of text product reviews. These reviews are usually accompanied by a 1 to 5 rating. This made these reviews extremely useful and fruitful in classification tasks to be able to predict sentiment of unseen data. Also with the presence of such good datasets, the assessment of classification algorithms for this task became possible [Wallin, 2014]. In this work, they gave comparisons and recommendations for sentiment analysis algorithms through working on one of the largest existing datasets (Movies & TV) from Amazon review collections.

In this chapter, we first explored the Movies & TV reviews dataset and its validity for the sentiment analysis problem, and then we used bag of words models and the TF-IDF algorithm to extract features from the text of the reviews. Next, we prepared the review for the classification problem. At the end, we evaluated the classification using F-measure and compared different classifiers on different data sizes. Then we did a similar study with a Book reviews dataset. Figure 5.1 present all the steps or workflow we followed for semantic analysis.

5.2 Dataset: Amazon Reviews

The dataset [Leskovec, 2013] that we are using in this problem is SNAP (Stanford Network Analysis Platform) Amazon reviews that were collected from amazon.com. With almost 35 million reviews, Amazon reviews is one of the biggest datasets for product reviews. These reviews were collected for 18 years up to March 2013. The dataset includes more enriching data like the rating of the review and product and user information. The rating is based on one to five stars where one means that the user did not like the product and five means that the user loved the product. In Figure 5.2 shows the details of the reviews star rating system.

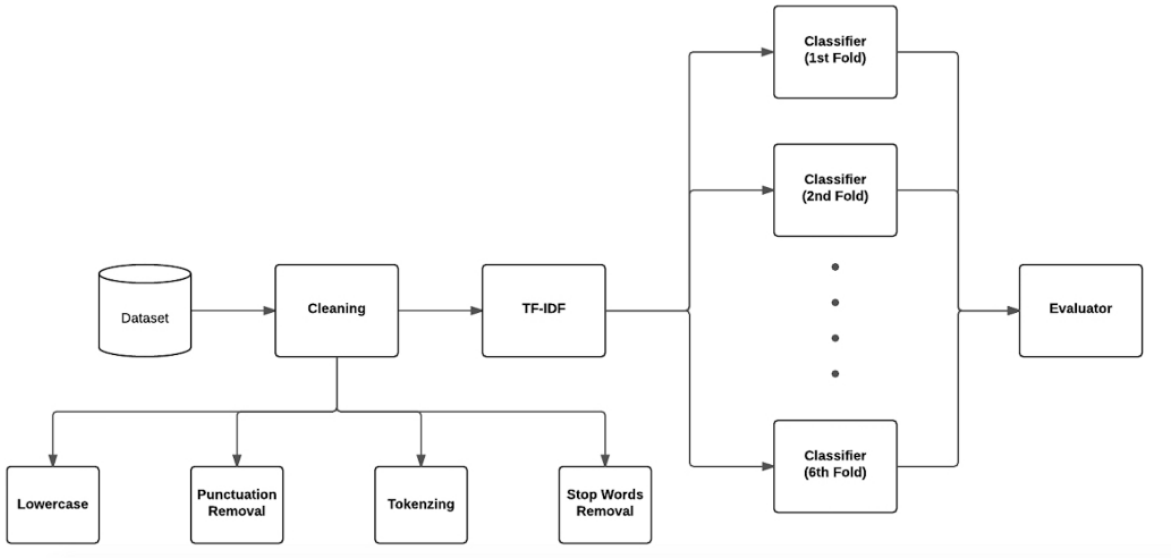

Figure 5.1: Semantic analysis workflow.

Star Level	General Meaning
☆	I hate it.
☆☆	I don't like it.
☆☆☆	It's okay.
☆☆☆☆	I like it.
☆☆☆☆☆	I love it.

Figure 5.2: Rating system for Amazon.com.

Amazon review dataset statistics contains:

- Number of reviews $34,686,770$
- Number of users $6,643,669$
- Number of products $2,441,053$
- Users with > 50 reviews $56,772$
- Median no. of words per review 82
- Timespan Jun 1995 - Mar 2013

The overall data spans over 30 categories with a total size of about 35 GB. From them we selected the two largest subsets: Books and Movies & TV.

5.3 Movies and Television

5.3.1 Dataset: Movies & TV Reviews

In our first experiment for sentiment analysis, we are using the "Movies & TV" dataset. The "Movies & TV" dataset consists of $2,408,693$ reviews from $70,385$ products. Size of this data

Features	Sample1	Sample1
product/productId	B00006HAXW	B0001Z3TLQ
product/title	Rock Rhythm & Doo Wop: Greatest Early Rock	By the Sea [VHS]
product/price	unknown	unknown
review/userId	A1RSDE90N6RSZF	A3421LTBSWSPXK
review/profileName	Joseph M. Kotow	KML
review/helpfulness	9/9	5/6
review/score	5.0	4.0
review/time	1042502400	1089417600
review/summary	Pittsburgh - Home of the OLDIES	A romantic zen baseball comedy
review/text	I have all of the doo wop DVD's and this one is as good or better than the 1st ones. Remember once these performers are gone, we'll never get to see them again. Rhino did an excellent job and if you like or love doo wop and Rock n Roll you will LOVE this DVD... !!	When you hear folks say that they do not make'em like that anymore, they might be talking about "BY THE SEA";. This is a very cool story about a young Cuban girl searching for identity who stumbles into a coastal resort kitchen gig with a zen motorcycle ...!

Table 5.1: Sample of Movies & TV reviews.

is about 2.92 GB compressed and 8.77 GB uncompressed. Each review includes the following information:

- product/productId: Asin, e.g. amazon.com/dp/B00006HAXW
- product/title: Title of the product
- product/price: Price of the product
- review/userId: Id of the user, e.g. A1RSDE90N6RSZF
- review/profileName: Name of the user
- review/helpfulness: Fraction of users who found the review helpful
- review/score: Rating of the product
- review/time: Time of the review (unix time)
- review/summary: Review summary
- review/text: Text of the review

Table 5.1 shows two samples of the Movies & TV dataset.

5.3.2 Experimental Setup

5.3.2.1 Target Variables

The goal of this experiment is classifying a product by positive review or negative reviews. For this reason we use review stars given in the range one to five as shown in Figure 5.2, where one indicates very low score and five indicates very high score. One way to provide better sentiment analysis is to represent the problem in only two classes: Positive and Negative. A negative review has a score less than 3 and a positive review has a score larger than 3. We chose to ignore the middle score due to its high neutrality.

5.3.2.2 Document Representation

In the data pre-processing, we used parallel computation where we divided the data into partitions to make the processing fast. We divided the file to 158 partitions. Each partition contained 546524 lines. We only used the first 74 partitions to make the final selected dataset. Those first 74 partitions gave me 3.2 GB of data. Below is summary of final selected dataset.

In the Movies & TV dataset we have $2,408,692$ reviews with:

- Number of users: $672,083$
- Number of products: $70,385$
- Users with more than a one review: $294,974$
- Median no. of words per review: 82.0
- Timespan Jun 1995 - Mar 2013
- Data size = 3.2 GB
- Avg# reviews/each product: 34.22.

5.3.2.3 Target Preparation

Before using the dataset in classification, we must prepare the target class. The reviews tend to be higher than lower causing a J shaped distribution. This distribution can make the classifier more tending to classify reviews as positive than negative. Figure 5.3 shows score distributed before normalization.

Online customers tend to give more positive reviews than negative reviews. This means customers are driven by purchasing bias and underreporting bias. A lot of care should be taken when handling such distribution to avoid having such bias in the classifier.

One method to handle unbalanced scores was to choose an equal portion from each score so that each score participates equally in the dataset. This will increase the relevance of underrepresented scores and decrease the relevance of over represented scores. This distribution is called rectangular distribution. This has a downside in that it decreases the size of the data, but leads to a much more balanced classification which leads to better classification and better results.

We used random sampling to avoid skewing the results. The count of products before this step is $70,385$ products and after this step reduced to $50,881$ products.

By using a J shape/rectangular distribution we reduced the data size from 3.2 GB to 1.2 GB.

Movies & TV dataset we have $675,070$ reviews with:

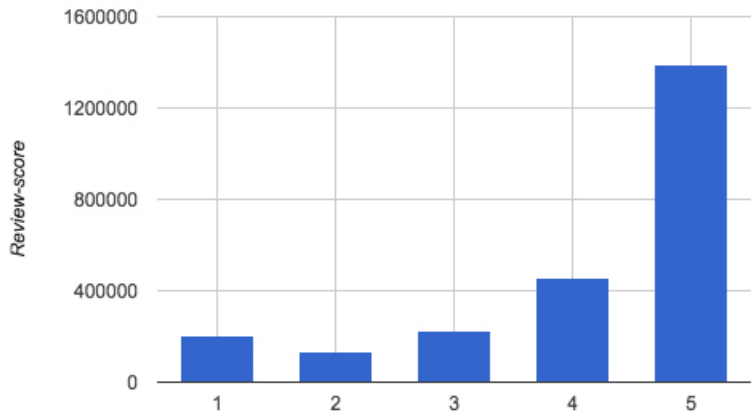

Figure 5.3: Score distribution before normalization.

- Number of users: $258,911$
- Number of products: $50,943$
- Users with > reviews: $79,590$
- Median no. of words per review: 97.0
- Timespan Jun 1995 - Mar 2013
- Data size = 1.2G
- Avg# reviews/each product: 13.27.

Now we have 13 reviews/product after normalization down from the 34 reviews/product before. normalization.

Reviews score are given in the range one to five, where one indicates a very low score and five indicates a very high score. One way to provide better sentiment analysis is to represent the problem in only two classes: Positive and Negative. We chose one and two ratings to present Negative samples. Four and five to represent Positive samples. We chose to ignore three due to its high neutrality and subjectivity.

After these two normalizations on Movies & TV dataset, we had $270,028$ positive and $270,028$ negative reviews.

5.3.2.4 Feature Extraction and Feature Selection

As previously said, Datasets have violations and removing these violations will produce more accurate results in any classification experiments made using any dataset. In this section, we describe the procedures necessary to remove these errors. We call the resulting corrected text categorization test collection Amazon-even.

Empty values can pollute the classification process. Removing them is a very critical step in the data preparation. The sentiment analysis is done on the review/text column. So, it cannot be null. There was only one row with empty review/text and it had to be deleted.

The different case of text affects the accuracy of the text Classification. Converting all text to lowercase so all words are treated the same wherever they are in the sentence solves

	A	B	C	D	E	F	G	H	I	J	K
1	product/productI	product/price	product/productI	product/title	review/helpfulnes	review/profileNar	review/score	review/summary	review/text	review/time	review/userId
2	B00004CM85	unknown	B00004CM85	For All Mankind [	11/29/2016	John Catsoulis	1	Misleading descr	Despite the Ama	1252108800	A23G24FDXKED34
3	B000AU9UYM	7.82	B000AU9UYM	Batman Begins (	8/37	James	1	Batman	I really enjoyed	1130025600	A2PXHT4JQXYH61
4	B0000YEE58	3.95	B0000YEE58	George C. Scott:	2/3/2016	dhqman	1	Movie Quit in mic	Watched movie t	1298160000	A27Z32X2G43XKF
5	B000085ROR	unknown	B000085ROR	The Musketeer (:	4/6/2016	Robert Law	1	The Worst Big Bu	Forget about Ba	1014249600	AJ2RU4VJ470IU
6	B001O0TMMM	unknown	B001O0TMMM	Thomas & Frien	12/17/2016	Gator Mom	1	Inappropriate for	In two of the epi	1167350400	A27ALKRE2RBQO3
7	B000085RPG	unknown	B000085RPG	The Texas Chain	2/16/2016	Rich	1	HORROR CULT	I always hear pe	962582400	A3D5RB2U0SIY7V
8	B00004RJED	unknown	B00004RJED	The Haunting [Vl	1/2/2016	unknown	1	EMBARASSING	This is the worst	940032000	unknown
9	B000085ROP	unknown	B000085ROP	The Last Unicorr	0/0	David Valentin	1	Incredible Movie	While in Germar	1079568000	A1183N27D7TW98
10	B000057EX8	unknown	B000057EX8	Super Mario Bro:	83/158	T. Sparks	1	My Childhood is	My father took n	1070064000	AEV1F9T5A362E
11	B000AU9UYM	7.82	B000AU9UYM	Batman Begins (	14/52	Leland Palmer	1	Should have be	for all the time w	1130284800	A2U5KHA9AJG9ZR
12	B0090XPGVG	10.32	B0090XPGVG	Constantine [Blu	6/13/2016	tastytravels	1	Rachel Weisz an	Unimaginative s	1121904000	A3OK5WB7IY25XH
13	B00005N5UZ	unknown	B00005N5UZ	The Mummy Ret	3/5/2016	Shane woodford	1	The Mummy Ret	The Mummy Ret	1000684800	A2UIZKPJPRSAW3
14	B000085RPG	unknown	B000085RPG	The Texas Chain	2/16/2016	Justin M Kirby	1	uneven	when i first watch	1071705600	A2N03TUL3EZZDD
15	B00005N5UZ	unknown	B00005N5UZ	The Mummy Ret	0/2	Barbara Joan Ma	1	The Mummy that	First of all I was	1005782400	AFIL3VVDX9FS3
16	B00008FG9K	unknown	B00008FG9K	Night With Lou R	6/25/2016	jazzie	1	When the smack	Lou dissapoints	987033600	A1VZ7Q9AC9MRZA
17	B00007A34S	2.99	B00007A34S	xXx (2002)	7/10/2016	B. Hill	1	head for the XXX	I've only ever wa	1031011200	A1WML695KKOBMZ
18	B00007A34T	2.99	B00007A34T	xXx [VHS] (2002	5/7/2016	unknown	1	Complete garbag	The only reason	1028937600	unknown
19	B003BV8I54	15.12	B003BV8I54	Porcupine Tree:	11/38	C. R. Johnson	1	Great band, HO	I don't get it. I lo	1283472000	A3PE4MZ0OFMPQ4
20	B000KX0HI2	12.98	B000KX0HI2	Hip Hop Uncens	3/4/2016	tjsmith	1	Hip Hop Uncens	i purchased this	1325635200	A2W4V5L7U07QEK
21	B000057EX8	unknown	B000057EX8	Super Mario Bro:	1/19/2016	Travis R. Wilson	1	The worst conso	I guess I was lik	1217894400	A1HYG113L4A2NX
22	B00007J5VT	11.8	B00007J5VT	8 Women (2002)	10/23/2016	unknown	1	Oh no - I had to	I agree with the	1051920000	unknown
23	B00004CM7Z	unknown	B00004CM7Z	Frankie and Johr	0/2	awol13	1	Frankie & Johnn	The DVD arrived	1294963200	A21HBRFRK2594Y
24	630018174X	35.95	630018174X	Seven Per Cent	1/23/2016	Helmut Schwarz	1	A Joke	One reviewer ac	1290456000	A3HF3Q8WQBBF2Y

Figure 5.4: Sample of dataset after normalization.

that. Punctuation signs and the stop words (the, a, am, I, are, at, etc) do not contribute in the classification process. Removing them early in the process produces better results. Tokenizing the text in which the documents are treated as strings and then partitioning the tokens into lists is the last step before feature selection.

In our problem, to analyze the text more, we chose to create our feature list from bigrams and unigrams, not only unigrams. This results in more features per review. For example, the sentence "I love NLP" has 3 unigrams (i, love, nlp) and 2 bigrams (I love, love NLP).

So for a three word sentence, unigrams generate three features while unigrams and bigrams generate five features. This increases heavily based on the length of the review text. For feature selection, we used Term Frequency and Inverse Document Frequency (TF_IDF) which were explained on Section 2.3.

5.3.2.5 Classification Methods and Evaluation Metrics

We used five algorithms for the classification task: Decision Trees (DT), Support Vector Machines (SVM), Random Forest (RF), Naïve Bayes (NB), and K-nearest neighbors (KNN). All of them were explained in Section 2.2.

In this experiment, we want to classify if a review is positive or negative. First we start by cleaning all the review text to remove punctuation and stop words changing all the letters to lowercase. Then we use Term Frequency-Inverse Document Frequency for feature extraction where each word is converted into a score representing its contribution in the positivity or negativity. After that, the classifier is trained using these features on 80% of the data where the remaining 20% of the data is left to evaluate the classifier.

5.3.3 Experimental Results

We divided the dataset into 80% training and 20% testing. Precision, recall and F-measure were used as the performance measures as defined in formulas (3.1), (3.2), and (3.3). We used a confusion matrix where we calculated the true positive, true negative, false positive and false negative from which we calculated the precision and recall. After that, we calculated the F-measure.

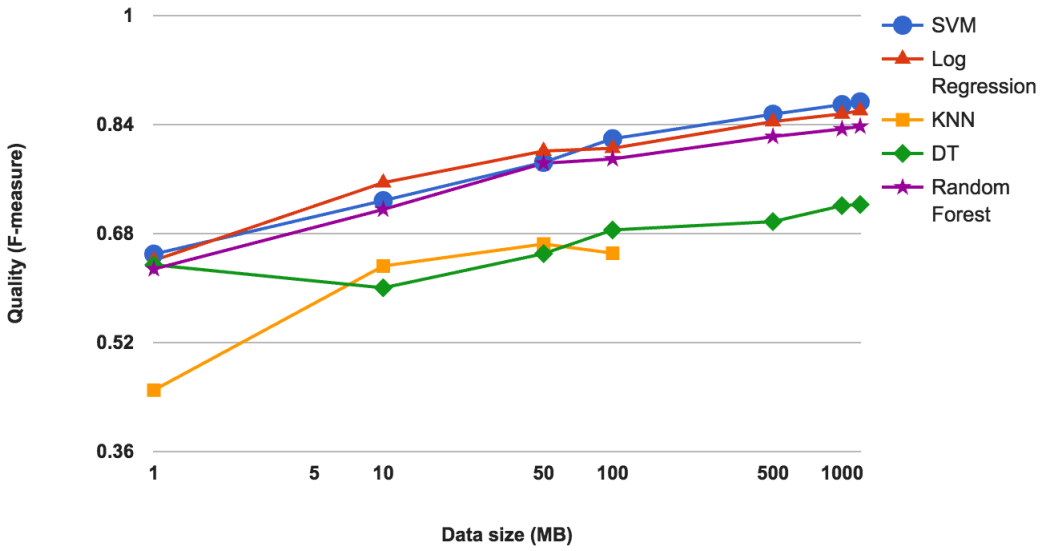

Figure 5.5: Comparison of algorithms quality on different data sizes.

In semantic analysis, we did not have to use any type of averaging (micro or macro) because we have a binary classification problem (two classes only). Averaging is only used for multi-classes.

In this section, we provide a comparison between the performances of classifiers when used on different sizes of data. We do not have Naïve Bayes due to the high number and sparsity of features.

5.3.3.1 Quality Comparison

Figure 5.5 shows the comparison of the quality of the algorithms when run on different data sizes. Through this experiment, we see that generally there is an increase of F-measure with increase of data size. SVM and Logistic Regression have very close scores, but SVM performs better with the increase of data.

Decision trees performed worse than them averaging around 60% F-measure. KNN started very low, but the problem with KNN is that the sparsity of the textual data made it very hard to train on the big sizes of data, consuming very huge amounts of data, as we can see later.

5.3.3.2 Time Efficiency Comparison

In this experiment, we wanted to see how much training time is required for each classifier on each data size. As we can see in the Figure 5.6, there is a general linearity in the sentiment analysis problem between the time needed to train and the data size. Also, it is clear that KNN stopped after 100 data size due to sparsity of the features as mentioned above.

5.3.3.3 Dominant Algorithms

Logistic Regression has the best quality on 10 MB, 50 MB data size. After that SVM has better quality and is a bit faster than Logistic Regression. KNN is the fastest algorithm on small data size to 50 MB data. In 10 MB data size, Logistic Regression and KNN are dominant. Figure

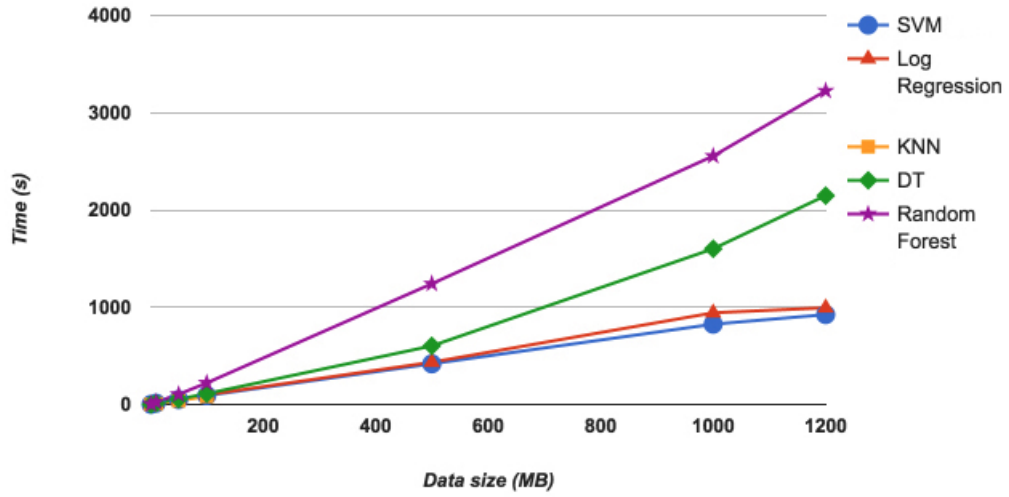

Figure 5.6: Time efficiency vs. data size.

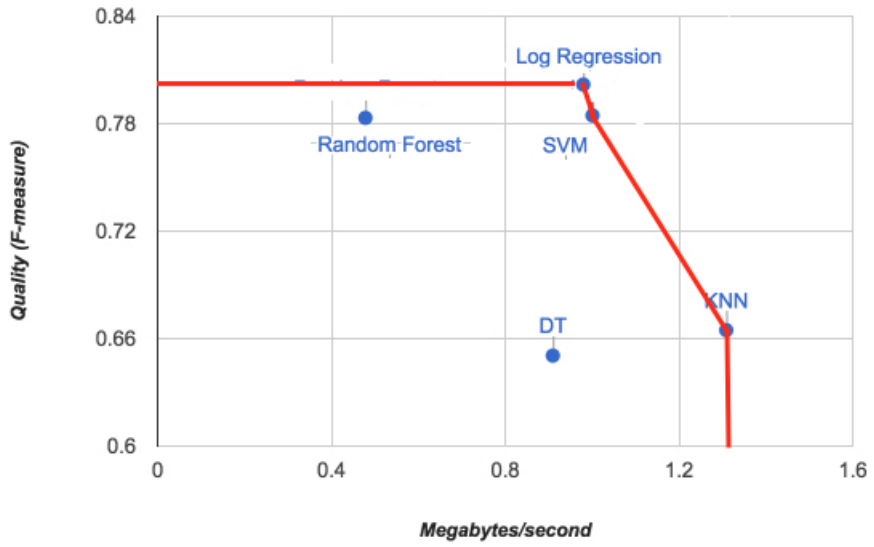

Figure 5.7: Dominant algorithms on Sentiment Analysis for 50 MB.

5.7 shows that Log regression, SVM, and KNN are the dominant algorithm in 50 MB data size. Therefore, depending on whether you choose speed or quality either algorithm can be selected as the dominant algorithm. Because of their inefficiency, to selecting Random forest and DT are poor choices.

On the other hand, KNN could not handle larger data sizes. Also, SVM performs better in quality and time in large data sizes as shown in Figures 5.8 and 5.9. Logistic regression and SVM are very close to each other in small data size to 500 MB data size. But by growing the data size this distance decreased.

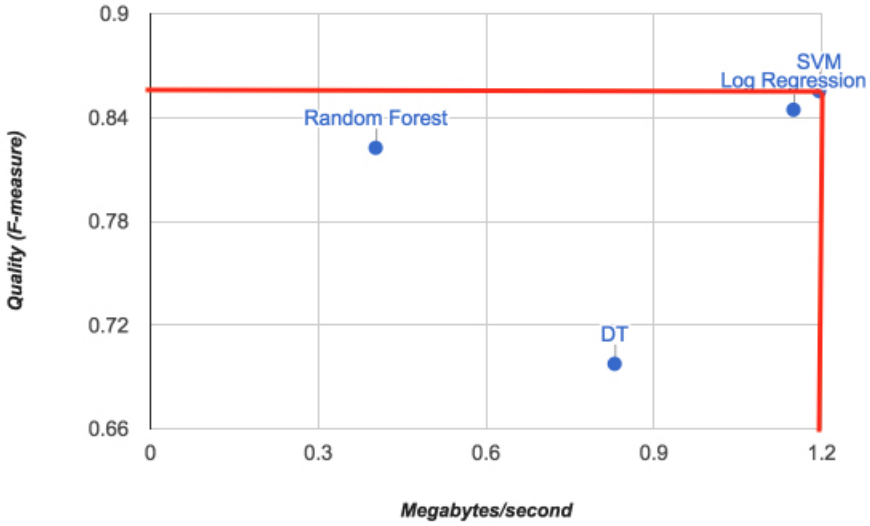

Figure 5.8: Dominant algorithms on Sentiment Analysis for 500 MB.

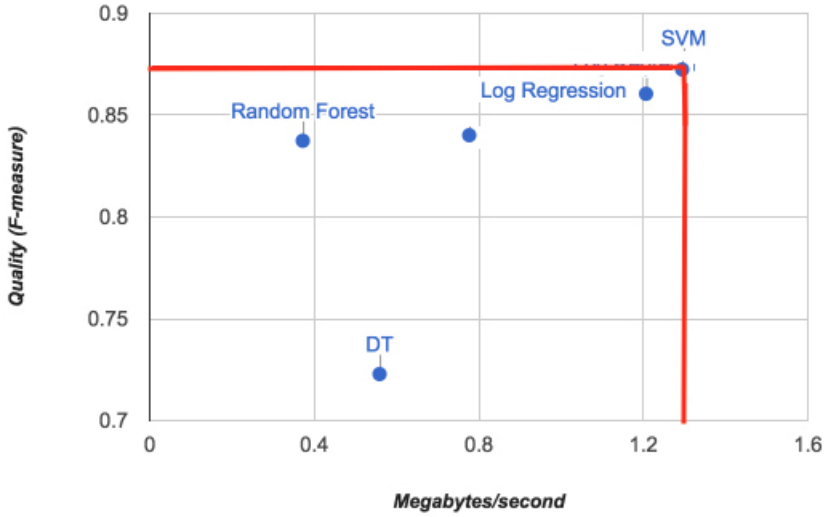

Figure 5.9: Dominant algorithms on Sentiment Analysis for 1.2 GB.

5.3.3.4 Overall Comparison

First, we show a standard comparison for some of the algorithms considering 500 MB data size. As shown in Figure 5.10, SVM and Log regression share almost the same the quality, with SVM being slightly better, while decision tree has a much lower quality.

On the training time scale, SVM has the lowest training time, due to its easy support vectors compared to Log regression mathematical equations. The Decision tree had a much higher training time due to the sparsity in the text features. As we explained in equation (3.4), performance can be considered as a combination of two factors: Quality and Time. Overall, SVM has the best performance for increasing data size. Logistic regression performance also increased

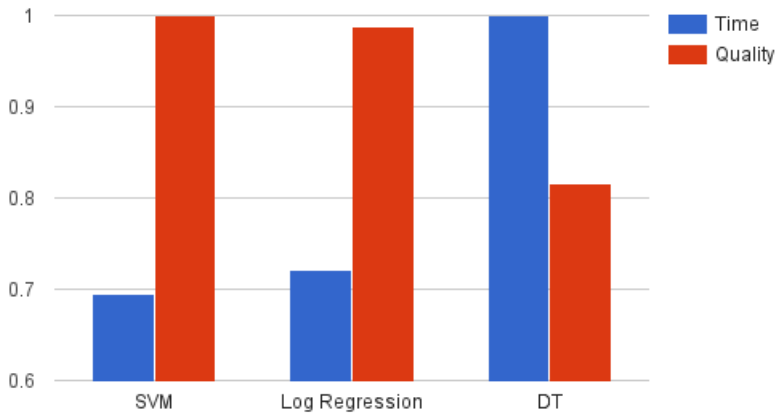

Figure 5.10: Comparing three algorithms in time and quality.

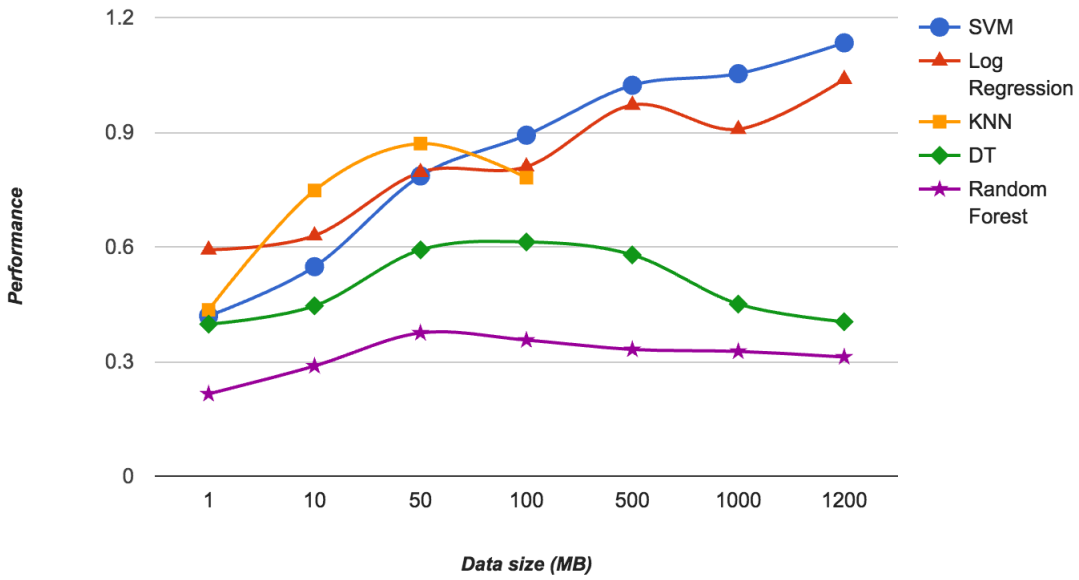

Figure 5.11: Performance comparison of the algorithms on different data sizes.

by increasing the data size. Random forest, Decision tree, and KNN all had performance decreases as the size of data increased. Logistic regression has better performance than SVM in semantic analysis when the data size is small as shown in Figure 5.11.

Finally, we obtain the graph in Table 5.2 that shows the best algorithm quality, speed, and performance on 1.2 GB data size. As we can see, SVM has better performance for big data. In small size of data KNN and Logistic Regression have a better performance than SVM.

Data size	Quality	Speed	Performance
1.2 GB	SVM	SVM	SVM
	Logistic Regression	Logistic Regression	Logistic Regression
	Random Forest	DT	DT
	DT	Random Forest	Random Forest

Table 5.2: Algorithms that had the best quality, speed, and performance on 1.2 GB data size.

5.4 Books

In this section, we use the Amazon reviews books dataset to test and study the classification algorithms. We use bag of words models and TF-IDF for feature selection and six algorithms for the classification task: Decision Trees (DT), Random Forest (RF), Naïve Bayes (NB), Support Vector Machines (SVM), K-nearest neighbors (KNN) and Logistic regression. Then we followed document classification process for target preparation, document representation, feature extraction and selection that was explained in Section 5.3.2. At the end of this section, we present the results of the experiments and discuss them.

5.4.1 Dataset: Books Reviews

The dataset we use for this experiment is part of the dataset from Amazon reviews that was divided into separate files for individual product categories and which have already had duplicate item reviews removed. Here we use the biggest data category, "Books". The size of this dataset is about 4.69 GB for the zipped file and about 14.39 GB for the unzipped file. The Book dataset consists of $12,886,488$ reviews with 10 features each from $929,264$ products. Other characteristics of this dataset are:

- Number of users: $2,588,991$
- Median no. of words per review: 92.0
- Timespan Jun 1995 - Mar 2013
- Average number of reviews per product: 13.87.

This dataset includes product metadata (brand, category information, descriptions, price, and image features), reviews (ratings, helpfulness votes, text), and links (viewed/bought graphs). [Leskovec, 2013]

As we explained earlier in chapter 6, the reviews were labeled based on one to five stars where one means a lower rate and five means a higher rate. Below is an example of features that we extracted for our experiment:

- reviewer ID - ID of the reviewer, e.g. A2SOAM1J3KNN3B
- asin - ID of the product, e.g. 0000012714
- reviewer Name - name of the reviewer
- helpful - helpfulness rating of the review, e.g. 2/3
- review Text - text of the review

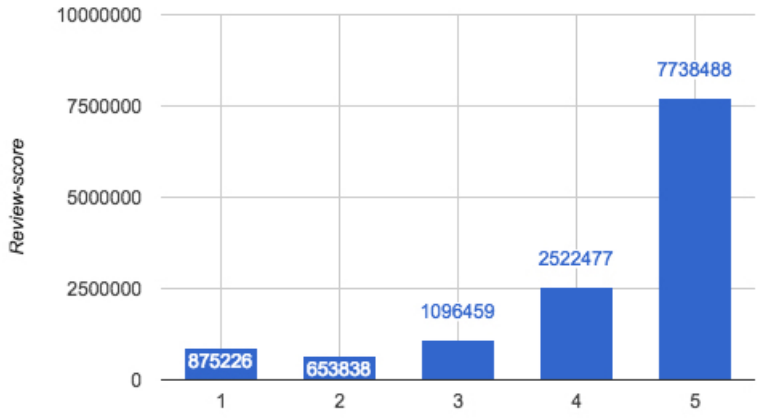

Figure 5.12: Score distribution of Amazon reviews.

- overall - rating of the product
- summary - summary of the review
- unix Review Time - time of the review (unix time)
- review Time - time of the review (raw)

Figure 5.12 shows the ratings distributed according to a J-shaped distribution with an overwhelming majority being positive reviews on the Book dataset.

Table 5.3 shows two sample of the Books dataset that include the following information:

5.4.2 Experimental Setup

5.4.2.1 Target Variables

In this section we used the same target variables as in Section 5.3.2.1. Figure 5.13 shows a sample of data after representing the problem in three classes: 0, 1 and 2 (positive, negative, middle).

	product/productId	review/score	review text	target15
2537385	0816152071	4.0	I think the book, "Circle Of Friends&quot...	1
70747	B000IZFOE4	1.0	The graphic and gorey descriptive writing by R...	0
1558184	B000MZSL5E	3.0	Not as impressive as some of his other work, D...	2
643508	B000MUQ2E6	1.0	I'd give it NO stars, but that isn't an option...	0
3082982	0929167058	5.0	I have been researching alternative health for...	1

Figure 5.13: Sample of data after target variables.

Features	Sample 1	Sample 2
product/productId	1882931173	0826414346
product/title	It is Only Art If It is Well Hung!	Dr. Seuss: American Icon
product/price	unknown	unknown
review/userId	AVCGYZL8FQQTD	A30TK6U7DNS82R
review/profileName	Jim of Oz	Kevin Killian
review/helpfulness	7/7	10/10
review/score	4.0	5.0
review/time	940636800	1095724800
review/summary	Nice collection of Julie Strain images	Really Enjoyed It
review/text	This is only for Julie Strain fans. It is a collection of her photos – about 80 pages worth with a nice section of paintings by Olivia. If you are looking for heavy literary content, this is not the place to find it – there is only about 2 pages with text and everything else is photos. Bottom line: if you only want one book, the Six Foot One is probably a better choice,...	I do not care much for Dr. Seuss, but after reading Philip Nel's book. I changed my mind–that is a good testimonial to the power of Rel's writing and thinking. Rel plays Dr. Seuss the ultimate compliment of treating him as a serious poet as well as one of the 20th century's most interesting visual artists, and after reading his book I decided that a trip to the Mandeville...

Table 5.3: Samples of books reviews.

Star Level	Ratings before cleaning	Rating after cleaning
★	875,226	875,226
★★	653,838	653,838
★★★	1,096,459	1,096,459
★★★★	2,522,477	2,522,477
★★★★★	7,738,488	7,738,474
Sum(1-5)	12,886,488	12,886,474

Table 5.4: Number of ratings per category before and after removing null reviews.

5.4.2.2 Target Preparation

Before using this dataset, we need to normalize a target class. A J-shaped distribution is presented in Figure 5.14.

Table 5.4 is obtained after removing empty reviews from our dataset. Before normalization, we have $(12,886,474,10)$ dimension of data. After normalization this dimension changed to $(3,269,190,10)$.

After normalization we have:

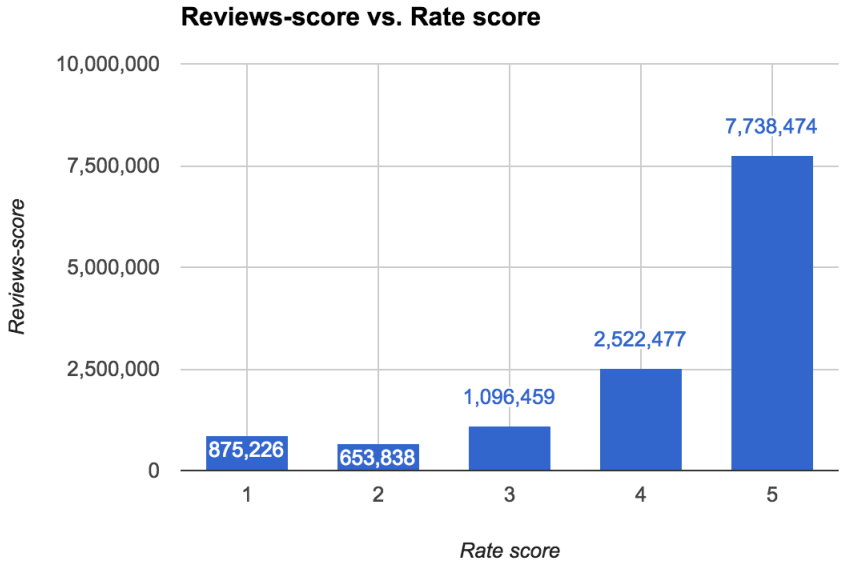

Figure 5.14: Score distribution before normalization.

- Number of reviews: $3,269,190$
- Number of users: $964,485$
- Number of products: $520,396$
- Median no. of words per review: 101.0
- Data size = 3.35 GB
- Average number of reviews per product: 6.28

After normalization, the number of ratings per categories rate is $653,838$ for all of them. We have $1,307,676$ positive and $1,307,676$ negative reviews.

5.4.2.3 Feature Extraction and Selection

We cleaned a dataset by removing all nulls and converted all the text to lowercase. Also, punctuation signs and the stop words were removed. At this point, tokenizing the text was done. For feature selection, we used TF-IDF which was explained in chapter 2.

5.4.2.4 Classification Methods and Evaluation Metrics

For classification, we use DT, Naïve Bayes, SVM, KNN, Random forest, and Logistic regression classification algorithms. These algorithms are explained in Chapter 2. For the classification evaluation, we used again a confusion matrix, precision, recall, and F-measure. We used 80% of the data for training and 20% of the data for testing in our experiment.

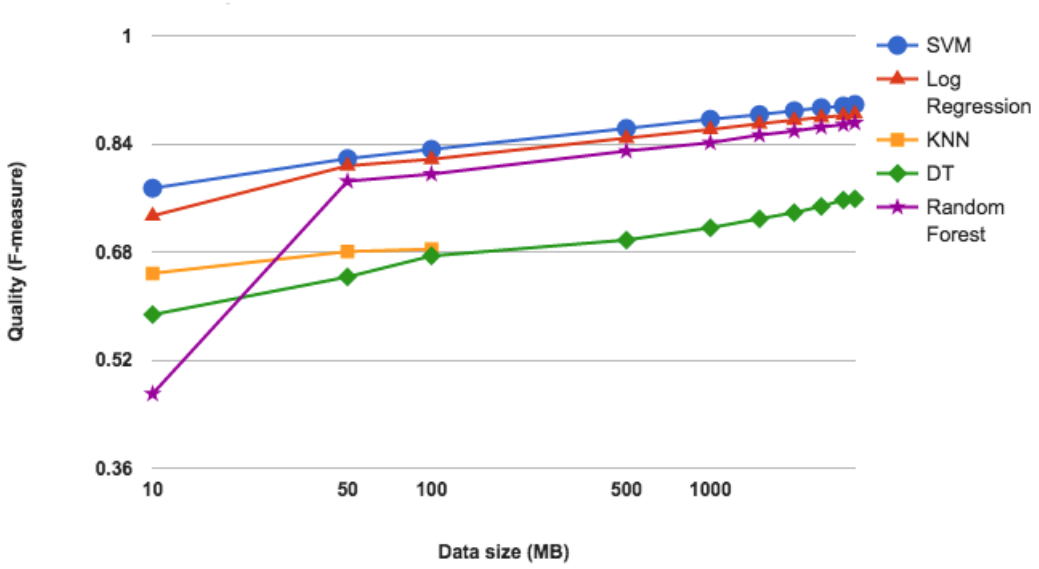

Figure 5.15: Comparison for quality on different data sizes.

5.4.3 Experimental Results

5.4.3.1 Quality Comparison

As we expected, with increasing data size, quality increased. RF started very slowly. SVM has shown to have had the best quality on big sizes of data. SVM and Logistic regression has the best quality on small data sizes. DT performed with less quality than the other algorithms most of the time.

5.4.3.2 Time Efficiency Comparison

By increasing data, running time will increase. RF is a lot more time consuming than the other algorithms. As we expected, after Random forest, Decision tree and then Logistic regression are the next most time consuming. SVM had the best running time among all algorithms for all the data sizes.

5.4.3.3 Dominant Algorithms

In this section, we determine the dominant algorithm for the 10 MB, 50 MB, 500 MB, 2GB and 3GB data sizes. SVM is the dominant algorithms in 10 MB data. In 50 MB, KNN and Logistic regression are dominant. SVM has better quality and running time on large data compared to the rest of the algorithms. Therefore, SVM is the dominant algorithm in large data sizes. As occurred for the Movies and TV dataset, Random forest and DT are not efficient algorithms.

5.4.3.4 Overall Comparison

Figure 5.20, shows the comparison of the classifiers on three main factors, quality, time, and data size, using the formula (3.4) to compare all three factors together. Logistic regression and KNN show the best performance on small data size. KNN crashed when increasing data. SVM

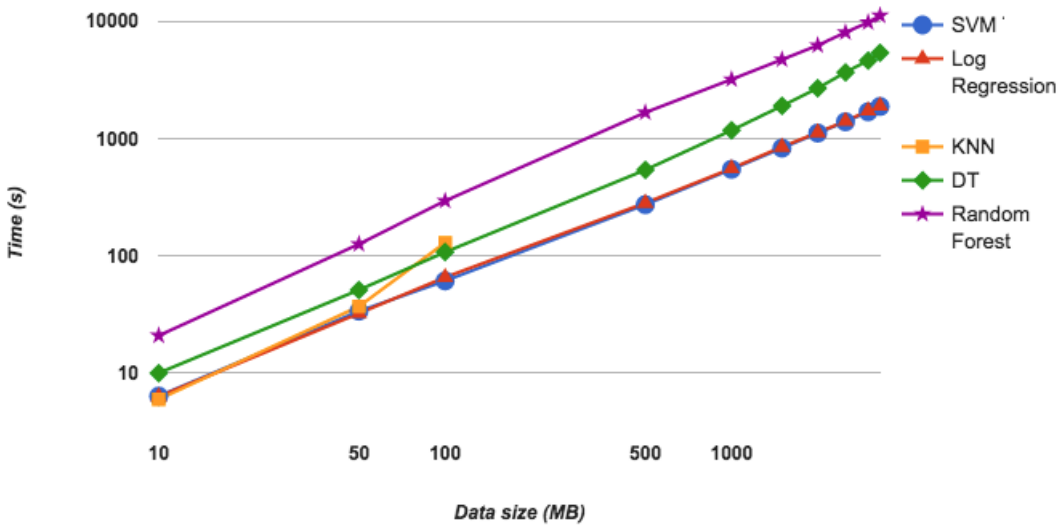

Figure 5.16: Comparison for time efficiency on different data sizes.

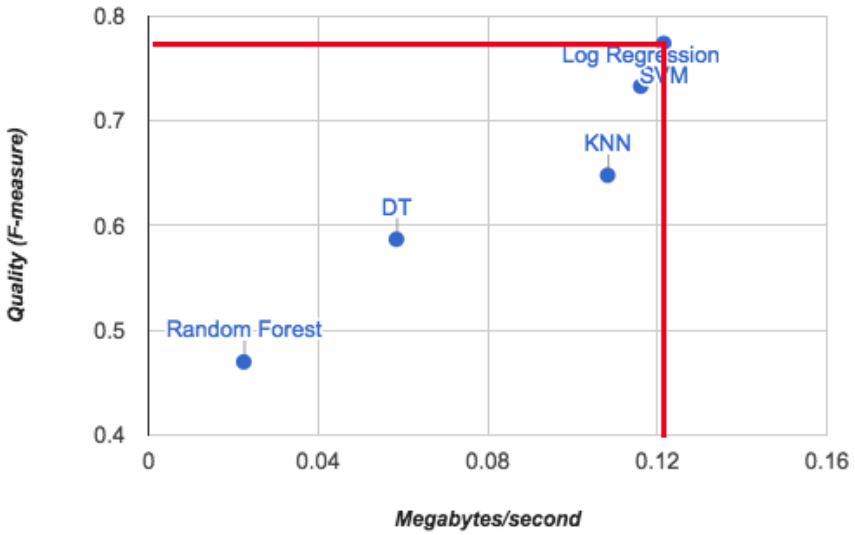

Figure 5.17: Dominant algorithms on Sentiment Analysis for 10 MB.

shows the best performance on large data sizes narrowly beating Logistic Regression. DT has shown stable performance. KNN and Random forest's performance drops down at 50 MB, but only Random forest after that increases again.

5.5 Discussion

In this chapter, we tested different algorithms on the sentiment analysis problem with two different subsets of the Amazon Reviews dataset. We got the same result in both experiments, corroborating our intuition that the results should not change because the subsets were similar.

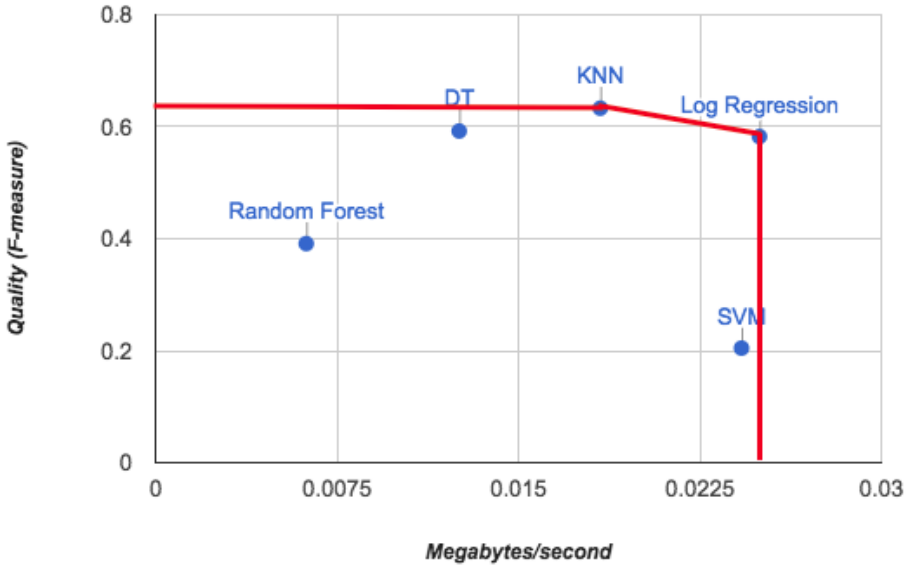

Figure 5.18: Dominant algorithms on Sentiment Analysis for 50 MB.

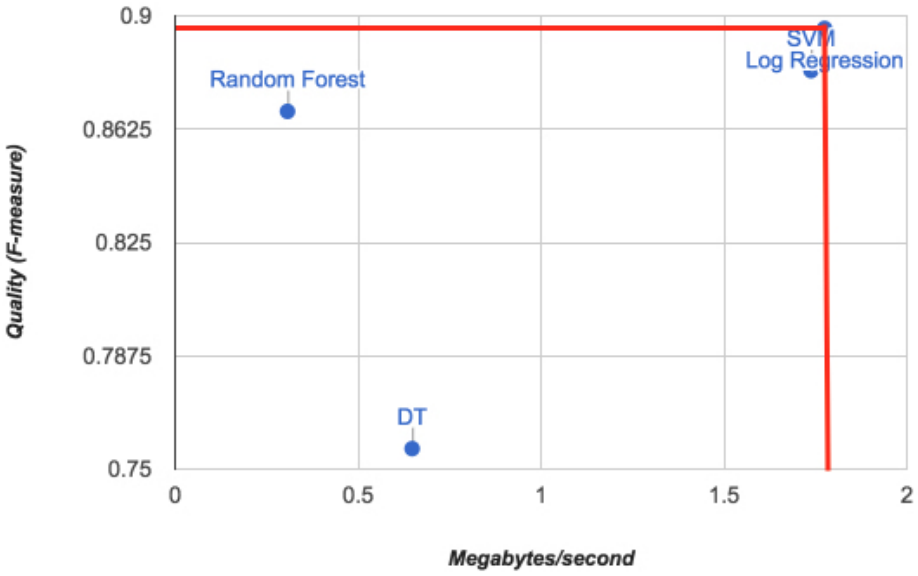

Figure 5.19: Dominant algorithms on Sentiment Analysis for 3 GB.

The results show that quality increased by adding more data. All the algorithms had the same learning curves in quality. However, our results show that in both cases increasing the training data does not always helps to improve the performance. Performance curves were different in both cases.

If we order the algorithms by quality, we have: SVM, Logistic regression, Random forest, and DT in larger than 100 MB data size. By considering only speed (fast-slow), the order changes to: SVM, Log regression, KNN, DT and Random forest in all data sizes. KNN crashed after several hours on 100 MB. Ordering by performance we have: SVM, Logistic regression,

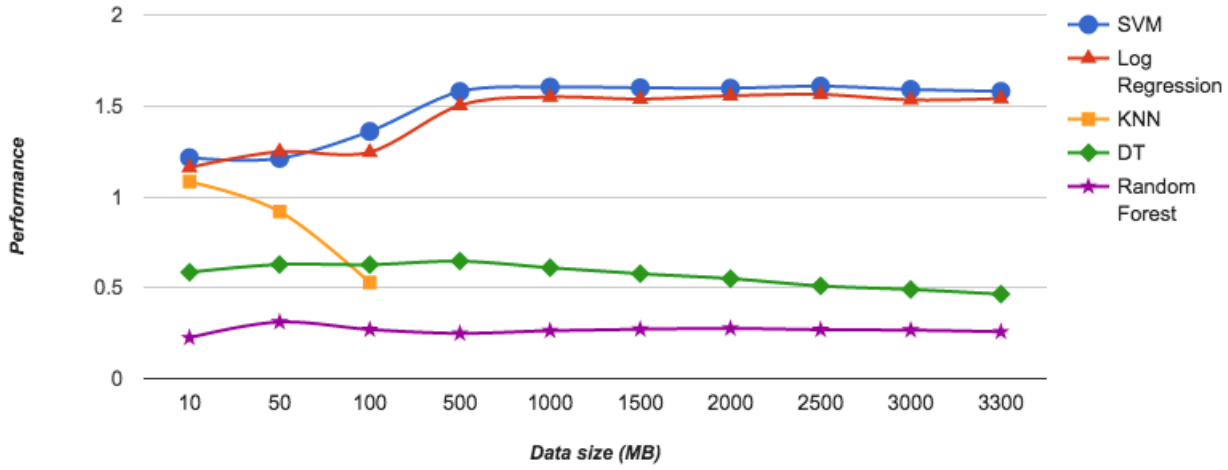

Figure 5.20: Performance comparison for different data sizes.

Data size	Quality	Speed	Performance
3.3 GB	SVM	SVM	SVM
	Logistic Regression	Logistic Regression	Logistic Regression
	Random Forest	DT	DT
	DT	Random Forest	Random Forest

Table 5.5: Algorithms that had the best quality, speed, and performance on 3.3 GB data size.

and KNN for less than 50 MB. The order for larger data sizes (more than 100 MB) is SVM, Logistic regression, Decision Trees, and Random forest.

Logistic regression has better performance in small data sizes. However, for larger data SVM shows better performance. Table 5.5 shows the best performing algorithm for quality, speed, and performance on the 3.3 GB data size (largest).

Regarding dominant algorithms, our results show that SVM is the dominant algorithm in data sizes over 50 MB. KNN, Logistic regression, and SVM are dominant in smaller data sizes.

In conclusion, if we are interested in quality or speed, we would recommend using SVM, Logistic regression or KNN for text classification or categorization in small data sizes. KNN could not handle data sizes larger than 100 MB and it crashed after 9 hours. For large data, the recommendation is to use SVM. Typically, SVM is not as favored for large-scale machine learning because the size of the dataset impacts the training complexity. Here we show SVM has the best performance (quality-running time) compared to the other algorithms that we have been investigated in this study. As occurred on the Movies and TV dataset, Random forest and DT are not efficient algorithms and hence they are not the dominant algorithms. From 500 MB to 3.3 GB quality did not change much. For instance, SVM's quality was 0.86 and 0.90 in 500 MB and 3.3 GB, and in 3.3 GB the running time was seven times higher than 500 MB. Therefore, it can be concluded that a 500 MB data size is a threshold point when performance no longer improves by adding more data.

Chapter 6

DOCUMENT CLASSIFICATION

6.1 Introduction

As the volume of information in digital form increases, the use of text categorization techniques aimed at finding relevant information becomes more necessary. Since the emergence of digital documents, there has been a requirement to automatically classify these documents. In recent years, the amounts of text documents have increased rapidly. This has made text classification a very important application and research topic [Ikonomakis et al., 2005, Khan et al., 2009, Khan et al., 2010].

The goal of categorization is to automatically assign an appropriate classification to each document that needs to be classified. Text classification is widely used in applications like finding answers to similar questions, finding whether an email is spam or legitimate email, finding the right category for a news piece, organizing documents into the correct folder, finding pages similar to a query, and much more. To build a classifier that can predict unseen documents, it must be trained using annotated documents assigned to each category that the classifier should be able to predict against [Wajeed and Adilakshmi, 2009]. For example, if you want to build a classifier that can classify whether a document is a sport, political, or fashion article, the classifier must see pre-classified articles in each of these three categories. Due to the nature of the classification process, several classification techniques are studied in the information retrieval field. For instance, both classification and information retrieval study how to get the best result for a certain query [Khan et al., 2009].

Due to the strong need for classification, classification is extensively studied and has advanced rapidly. Machine learning approaches are extensively used in classification whether supervised, semi-supervised, or unsupervised learning. Machine learning approaches are widely used and experimented on for classification. These approaches showed different results for each use case in classification. The most popular machine learning approaches used in classification are: Support Vector Machines (SVM), Decision Trees, K-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Bayesian Classifier, and Latent Semantic Analysis (LSA) [Gandhi and Prajapati, 2012, Ikonomakis et al., 2005, Khan et al., 2009, Khan et al., 2010].

A lot of research focuses on comparing the performance of different classifiers to choose the best classifier for the required task. Performance can be evaluated using different evaluations. Performance can be evaluated by training efficiency which is the time used by the classifier to learn the training data. Applications that learn online must have high training efficiency. Performance can also be evaluated using classification effectiveness, which is how correct the classifications are. Applications are critical like cancer classifiers must have high classification effectiveness. Performance can also be evaluated using classification efficiency which is the

time used by the classifier to classify a document. Applications that classify on the go must have high classification efficiency. To measure the performance correctly, the annotated documents are separated into training and testing set. The classifier is trained using the training set then tested using the testing set to prevent the test from suffering from overfitting.

Classifier performance can be improved on different axes like: Training the classifier to predict more documents correctly, Finishing the learning phase faster, Classifying the unseen documents faster, or requiring a smaller number of annotated documents for the training phase.

In this chapter, we focus on supervised learning. We used Decision Trees, Logistic Regression, K-nearest neighbors, Support Vector Machines, and Bayesian classifier and provide different experiments to compare between the classifiers on different aspects and find the best single-label classifier using these different axes:

1. By comparing quality between five different classifiers using different sizes of RCV1 dataset.

2. By comparing running time five different classifiers using different sizes of RCV1 dataset.

3. By comparing all three factors, quality, data size, and time.

In the introduction of this chapter we explained the area of text categorization. This chapter is divided into two different experiments; News classification and Patent classification. In each section, we present some text classification algorithms, and then we explain a dataset and the contributions of this work, and describe the outline of this part of this dissertation. We explain each step as we go through it.

6.2 Algorithms

In News classification, we used the following classifiers such as:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbors
- Support Vector Machines
- Naïve Bayes Random forest

The same classifiers are used in the Patent classification, but we used Random forest instead of Naïve Bayes for scalability reasons. In the Section 2.2, we looked at these algorithms and the differences they have.

6.3 News Classification

For News classification, we used the same dataset as used for NER in Chapter 4. Nevertheless, as here we use that dataset for a different task, we give more details. We classified documents using their content into predefined classes following our proposed methodology.

6.3.1 Dataset: News

Datasets are collections of pre-classified documents. They are essential to develop and evaluate a text classification (TC) system, that is, to train the system and then to test how well it behaves, when given a new document to classify.

The dataset consists of a set of documents, along with the category or categories that each document belongs to (target). In a first step, called the training phase, some of these documents (called the training documents) are used to train the TC system, by allowing it to learn a model of the data. Afterwards, in a step called the test phase, the rest of the documents (called the test documents) are used to test the TC system, to see how well the system behaves when classifying previously unseen documents.

To allow for a fair comparison between several TC systems, it is desirable that they are tested in equivalent settings. With this goal in mind, several data collections were created and made public, generally with a standard train/test split, so that the results obtained by the different systems can be correctly compared.

Some of the publicly available collections are more used than others. In the TC field, and in the single-label sub-field the most commonly used collections are the 20-Newsgroups collection, the Reuters-21578 collection (RCV1), and the WebKB collection.

Comparing the datasets together, RCV1 shows better documents in both quantity and structure. For example. RCV1 consists of more than 800k documents ($806,791$ for RCV1-v1, and $804,414$ for RCV1-v2) which is 35 times as many as Reuters-21578 and 60 times as many datasets with reliable coding and 2.5 times as many as OHSUMED [Lewis et al., 2004] where the later has another disadvantage that it does not contain the full text of documents. This disadvantage is not present in RCV1 where the full text. RCV1 documents also have unique document ID, which will be useful in our work [Lewis et al., 2004].

For this section, we chose the RVC1 dataset. RCV1 is an archive of over $800,000$ manually categorized newswire stories recently made available by Reuters, Ltd. for research purposes. Use of this data for research on text categorization requires a detailed understanding of the real world constraints under which the data was produced" [Lewis et al., 2004]. The data comes in 365 folders each containing thousands of XML files representing news articles in each day of the year. One of the XML news files is shown in Figure 6.1.

We needed a parser to convert XML to a format that can be easily used in analysis. The total amount of data is $806,791$ articles. Figure 6.2 is a sample of RCV1 after parsing to a CSV format.

6.3.2 Experimental Setup

6.3.2.1 Target Variables

RCV1 documents contain three variables that can be categorized against. These variables are: Industries, Regions, and Topics. From these variables, we used Topics variable because it variable is more completed than the other variables, less missing value and we can use content of news for subject classification. [Lewis et al., 2004]

2286newsML.xml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="2286" id="root" date="1996-08-20" xml:lang="en">
<title>MEXICO: Recovery excitement brings Mexican markets to life.</title>
<headline>Recovery excitement brings Mexican markets to life.</headline>
<byline>Henry Tricks</byline>
<dateline>MEXICO CITY</dateline>
<text>
<p>Emerging evidence that Mexico's economy was back on the recovery track sent Mexican markets into a buzz of excitement Tuesday, with stocks
closing at record highs and interest rates at 19-month lows.</p>
<p>&quot;Mexico has been trying to stage a recovery since the beginning of this year and it's always been getting ahead of itself in terms of
fundamentals,&quot; said Matthew Hickman of Lehman Brothers in New York.</p>
<p>&quot;Now we're at the point where the fundamentals are with us. The history is now falling out of view.&quot;</p>
<p>That history is one etched into the minds of all investors in Mexico: an economy in crisis since December 1994, a free-falling peso and
stubbornly high interest rates.</p>
<p>This week, however, second-quarter gross domestic product was reported up 7.2 percent, much stronger than most analysts had expected.
Interest rates on govermnent Treasury bills, or Cetes, in the secondary market fell on Tuesday to 23.90 percent, their lowest level since Jan.
25, 1995.</p>
<p>The stock market's main price index rallied 77.12 points, or 2.32 percent, to a record 3,401.79 points, with volume at a frenzied 159.89
million shares.</p>
<p>Confounding all expectations has been the strength of the peso, which ended higher in its longer-term contracts on Tuesday despite the
secondary Cetes drop and expectations of lower benchmark rates in Tuesday's weekly auction.</p>
<p>With U.S. long-term interest rates expected to remain steady after the Federal Reserve refrained from raising short-term rates on Tuesday,
the attraction of Mexico, analysts say, is that it offers robust returns for foreigners and growing confidence that they will not fall victim
to a crumbling peso.</p>
<p>&quot;The focus is back on Mexican fundamentals,&quot; said Lars Schonander, head of researcher at Santander in Mexico City. &quot;You have
a continuing decline in inflation, a stronger-than-expected GDP growth figure and the lack of any upward move in U.S. rates.&quot;</p>
<p>Other factors were also at play, said Felix Boni, head of research at James Capel in Mexico City, such as positive technicals and economic
uncertainty in Argentina, which has put it and neighbouring Brazil's markets at risk.</p>
<p>&quot;There's a movement out of South American markets into Mexico,&quot; he said. But Boni was also wary of what he said could be &quot;a
lot of hype.&quot;</p>
<p>The economic recovery was still export-led, and evidence was patchy that the domestic consumer was back with a vengeance. Also, corporate
earnings need to grow strongly to justify the run-up in the stock market, he said.</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
  <code code="MEX">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
</codes>
<codes class="bip:topics:1.0">
  <code code="E11">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
  <code code="ECAT">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
  <code code="M11">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
  <code code="M12">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
  <code code="MCAT">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed" date="1996-08-20"/>
  </code>
</codes>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-08-20"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="MEXICO CITY"/>
<dc element="dc.creator.location.country.name" value="MEXICO"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

Figure 6.1: Sample of XML file.

6.3.2.2 Topic Codes

RCV1 dataset [Lewis et al., 2004] comes with file "topic_codes.txt" which acts as a legend for topic codes. This file contains 126 codes, while only 103 codes are present in the dataset. All the 103 codes appear more than one time. These codes range from five occurrences for GMIL (MILLENNIUM ISSUES) to 374, 316 for CCAT (CORPORATE/INDUSTRIAL).

The structure of the codes is designed to provide two useful features:

1. Create a hierarchy to support automated assignment of more general topic codes.

2. Group related codes. These groups have an alphanumeric sort order to support manual lookup. For example: The code C31 (MARKETS/MARKETING) is the ancestor of the code C311 (DOMESTIC MARKETS). Also the code C311 also appears near related codes, such as C32 (ADVERTISING/PROMOTION).

The hierarchy of topic codes automated assignment can be found as follows:

1. Use the following four codes as single letters as follows: CCAT as C, ECAT as E, GCAT as G, and MCAT as M. These four letters are the parent root of the tree.

index	date	headline	itemid	region	text	title	topics
3	1997-08-13	PRESS DIGEST - VIETNAM - AUG 13.	798108	VIETN	These are some of the leading stories in the o...	VIETNAM: PRESS DIGEST - VIETNAM - AUG 13.	GCAT
5	1997-08-13	Fleet to buy Columbia Management- USA Today.	798322	USA	Fleet Financial Group Inc has agreed to buy pr...	USA: Fleet to buy Columbia Management- USA Today.	C18 - C181 - CCAT
6	1997-08-13	Stocks, dollar fell but bonds rose after econo...	796550	USA	Stocks fell on Wednesday though bond prices ro...	USA: Stocks, dollar fell but bonds rose after ...	M11 - M12 - M13 - M132 - M14 - M142 - M143 - MCAT
7	1997-08-13	Israel tears down five illegally built Arab ho...	797319	ISRAEL	Israeli Jerusalem municipality workers tore do...	ISRAEL: Israel tears down five illegally built...	GCAT - GPOL
8	1997-08-13	INDICATORS - EGYPT - UPDATED AUG 13.	797271	EGYPT	---INTEREST RATES--- \t\t\t\t\t Latest Date...	EGYPT: INDICATORS - EGYPT - UPDATED AUG 13.	E71 - ECAT
9	1997-08-13	Manugistics sets 1.9 mln shr offering.	795609	USA	Manugistics Group Inc said Wednesday it has la...	USA: Manugistics sets 1.9 mln shr offering.	C17 - C171 - CCAT
10	1997-08-13	TAIWAN MONEY RATES END DOWN ON FUND INJECTION.	796455	TAIWAN	Taiwan's money rates closed sharply lower on W...	TAIWAN: TAIWAN MONEY RATES END DOWN ON FUND IN...	M13 - M131 - MCAT

Figure 6.2: Sample of data after parsing.

2. Remove the minimal suffix till you find another code to find the parent of a code.

This hierarchical structure is not the only one. Another version can be formed for example: we can create another hierarchy by adding 13 artificial codes: C1-C4, E1-E7, G1, M1 and repeat the previous method.

Editors were allowed to assign any of the 103 topic codes, not just the leaves topic codes. They were asked to choose the most applicable and informative topic code wherever it's in the hierarchical structure. The result of this is that documents can have multiple categories: The most applicable code and all its ancestors. The ancestors' categories act as "Other" categories.

6.3.2.3 Document Representation

Text representation is the important aspect in documents classification. Documents are complex and hard to handle in their full text format. A pre-processing step is required to make them easier to handle. Documents representation is a pre-processing technique used to transform the documents from their complex text format to a document vector. This document vector consists of word features (vector of term weights). Each word is represented as a weight: for example, the count of occurrences of the word. One challenge in text classification is the high dimensionality of the textual features. Most of the times, the number of these features is greater than the documents present in the training data. "Text classification is an important component in many informational management tasks, however with the explosive growth of web data, algorithms that can improve the classification efficiency while maintaining accuracy are highly desired" [Khan et al., 2009].

Figure 6.3 shows the steps in document classification.

6.3.2.4 Target Preparation

Before using the dataset in classification, we must prepare the target class first. In this work, the target class is the Topics field. As we discussed earlier, Topics has 126 unique code. Figure 6.4 shows the frequency of each topic.

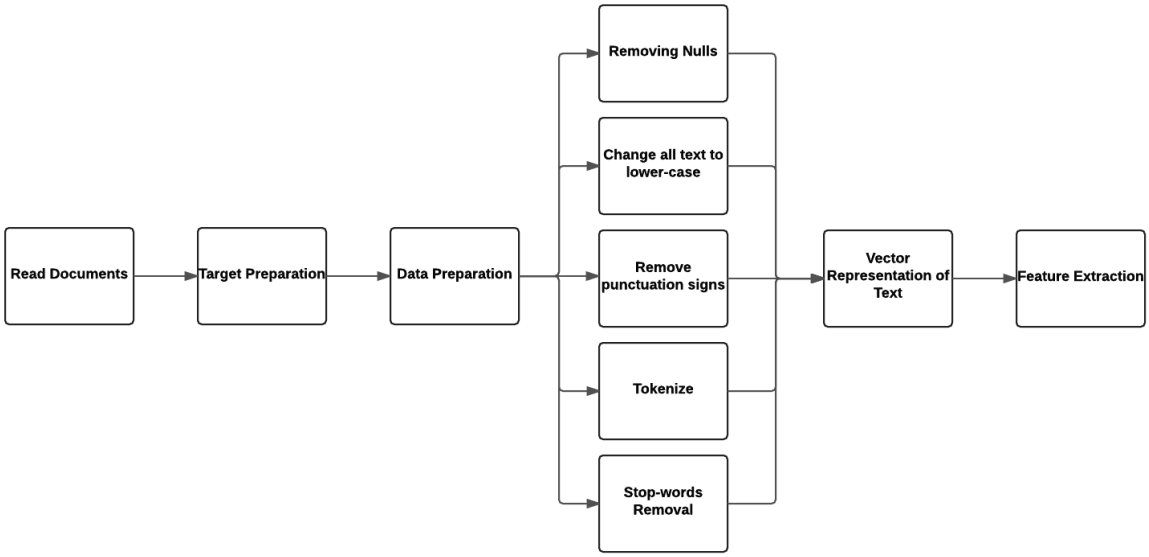

Figure 6.3: Document classification process.

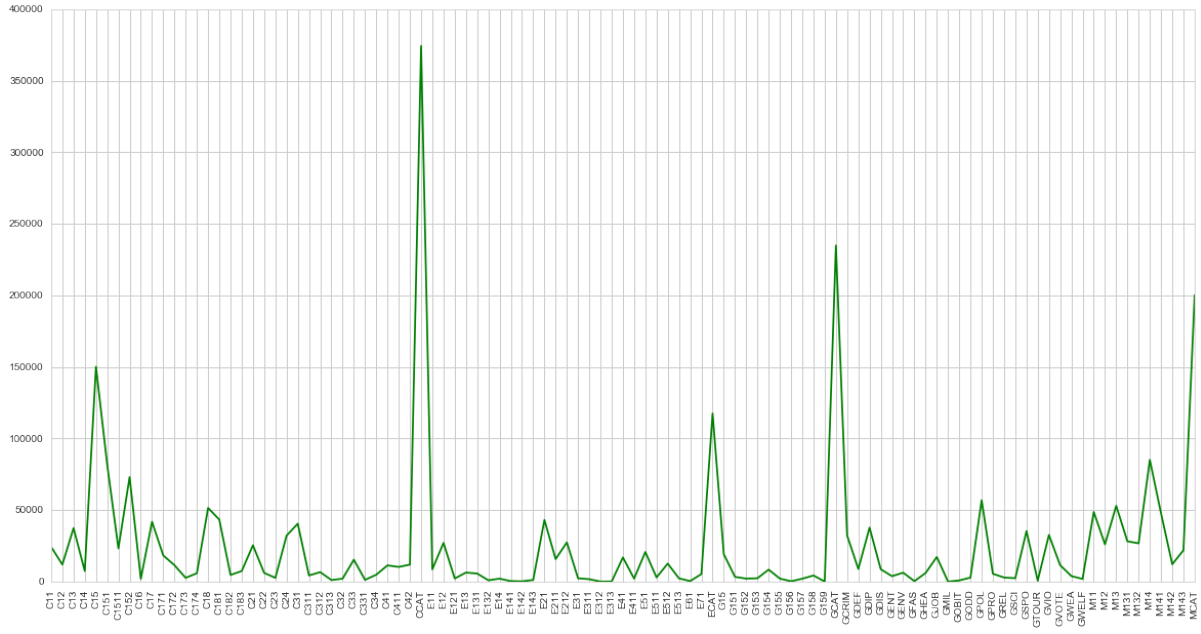

Figure 6.4: Frequency of topics in RCV1.

Some topics are used more than others and we captured the five most used topics (Count > $100,000$) and used them as our target in the classification process. However, there are topics that are used in multiple news articles. So, we chose the topic of the article that is used the least among our five most used topics. Table 6.1 shows the count of articles after target preparation.

As previously noted, datasets have violations and removing these violations will produce more accurate results in any classification experiments made using RCV1. In this section, we describe the procedures necessary to remove these errors. We call the resulting corrected text categorization test collection RCV1-v2.

Empty values can pollute the classification process, removing them is a very critical step in

Topic	Topic Name	Article Counts
C15	Performance	149,124
CCAT	Corporate/Industrial	163,804
ECAT	Economics	117539
GCAT	Government/Social	189,048
MCAT	Markets	184,901
Total		804,416

Table 6.1: Selected topics and article counts.

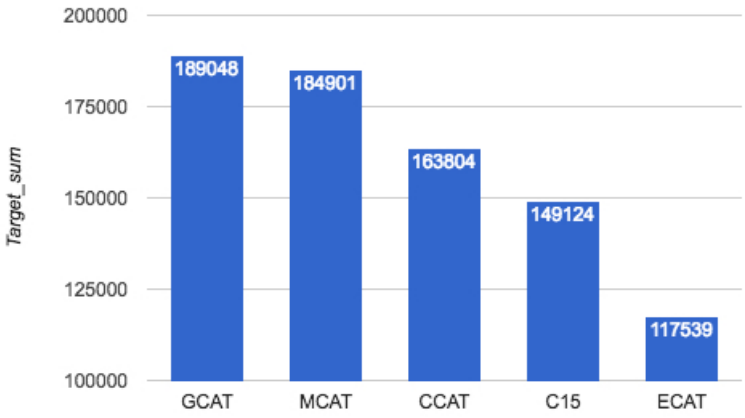

Figure 6.5: Frequency of selected topics in our dataset.

the data preparation. Removing nulls can happen by removing the entire column or populating the empty value. In our work, columns industry $(454, 983)$, dateline $(50, 744)$, byline $(697, 360)$ have huge number of nulls and probably will pollute the data if used in classification. They do not have a specific trend so the best solution is to remove them. Also, there were 14 articles with empty text or title, which must be removed as the features, which will be extracted from them, as we will discuss later. Also 22 articles were found with an empty region, but with region data present in the title column, so the best solution is to replace the null in empty regions with the region from the title column.

The different case of text affects the accuracy of the text classification. Converting all text to lowercase allows all words to be treated the same regardless of where they are in the sentence.

Punctuation signs and the stop words (*the*, *a*, *am*, *I*, *are*, *at*, etc.) do not contribute to the classification process. Removing them early in the process produces a better result.

Tokenizing the text that the documents are treated as strings and then partitioned into lists is the last step before feature selection.

1. Delete rows with empty text or title, since the features will be extracted mainly from them, So they cannot be empty (total count of deleted rows = 14)

2. Explore rows with empty topics

    - The rows are diverse and random, no pattern can be withheld to substitute for the lack of topics data.

- Since topics will be the target data, it cannot be null.

3. Delete rows with empty topics (total count of deleted rows = $2,363$)
4. Explore rows with empty regions
   - Region data is found in title column
5. Replace the null in empty regions with the region from the column with respect to the region codes.
6. The features matrix after cleaning is: $804,416$ rows $\times$ 8 columns

6.3.2.5 Feature Selection

Feature selection is the next step after feature extraction. Feature selection is a very important step in pre-processing of text classification due to the high dimensionality problem of textual features. Feature selection is only good if it considers the nature of classification algorithm and the domain. Feature selection in text classification is done by keeping the most important words for the classification through evaluating each word and giving them scores. The selected words should provide meaning and a better understanding for the data and learning process. The challenge facing feature selection is the presence of noise and stop words and a lot of non-meaningful words which can reduce the classification efficiency strongly [Khan et al., 2009]

Feature selection in machine learning is one of two types: Wrappers and filters. Wrappers create feature subsets and train the classifier on each of these subsets to be evaluated using the classification accuracy. This process is expensive and time consuming when the number of features is high which is the case in text documents. Therefore, wrappers are not the best case for text classification [Khan et al., 2009].

On the contrary, Filters use evaluation metrics to evaluate the feature ability to differentiate between each class. This means that filters are independent on the learning algorithm. In text classification, a document can partially be classified into many classes. Therefore we need to find the best class that matches a document. Term frequency inverse document frequency (TF-IDF) is is a commonly used feature selection algorithm for text classification. TF-IDF measures how unique the word is in the document and how important this word is in this document against all the corpus. Thus, TF-IDF converts the text data into Vector Space Model (VSM) reducing the complexity of the data and decreasing the noise of the text features helping to increase the accuracy of the classification [Khan et al., 2010, Khan et al., 2009, Frakes and Baeza-Yates, 1992].

For feature selection, we used Term Frequency and Inverse Document Frequency (TF_IDF) which are explained on Section 5.3.1.6-8.

6.3.3 Experimental Results

For our quality evaluation, we used Precision, Recall and F-measure that were explained in Chapter 3. Because we have more than one class, we need to combine multiple quality measures into one quantity. Macro Averaging is to compute quality for each class (precision and recall) and then average all of them. Micro Averaging is to collect decisions for all classes, compute the contingency table, and evaluate. For the evaluation we use the simple case of 80% of the data for training and 20% of the data for testing. Our experiment is divided into three parts as described below, following our methodology.

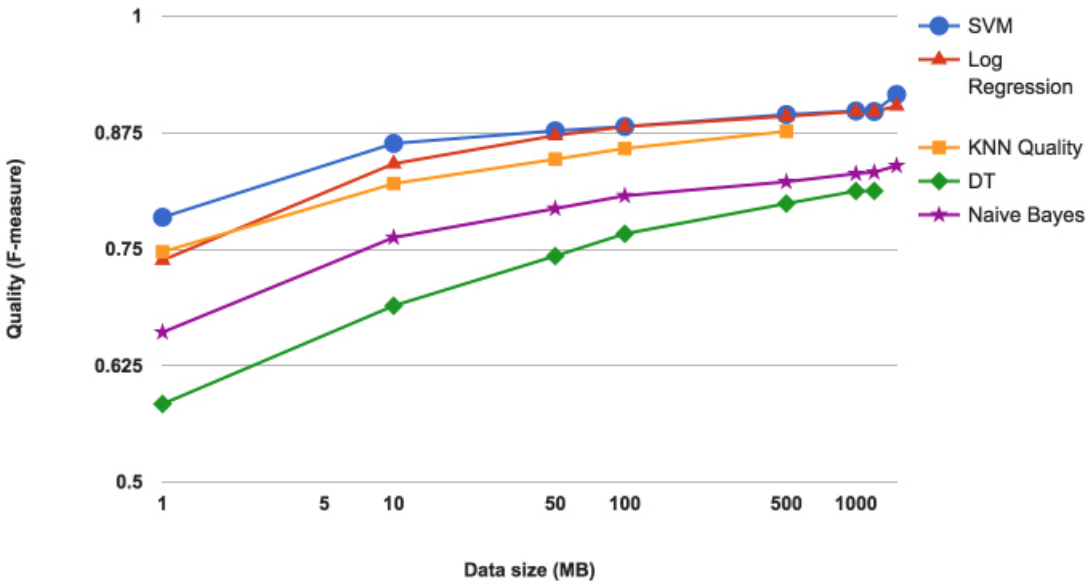

Figure 6.6: Quality vs. Data size.

6.3.3.1 Quality Comparison

After training, we used F-measure to measure the quality of the classifiers as explained in 3.2.3. We choose F-measure because accuracy is not always a good performance metric and usually high accuracy can be achieved by predicting the most common class. On the other hand, F-measure captures both sensitivity and specificity.

We vary the amount of training data and compare the classification quality of each classifier, the result shows in Figure 6.6.

SVM performs better on small sizes of data; however, when there is a big amount of data. Logistic Regression jumps to be equally efficient. Decision trees and Naïve Bayes perform badly compared to SVM and Logistic Regression. KNN is near to Logistic Regression on small and medium data sizes, but the problem with KNN is that the sparsity of the textual data made it very hard to train on the big sizes of data consuming very huge amount of data as we can see later.

6.3.3.2 Time Efficiency Comparison

Time factor is very critical in text classification. Since many applications are using text classification in real time operations, classifiers that take too much time will not be efficient for the task. For that we measured running time for each classifier on the different data sizes. The result of this experiment shows in Figure 6.7.

As we mentioned earlier, KNN taking highest time due to sparsity. Also the Decision tree, took the second place for time consumed to make classifications. SVM, Logistic Regression and Naïve Bayes are very close to each other in a very acceptable range to real time operations. Even on a big size of data, they could do the classification task in milliseconds range.

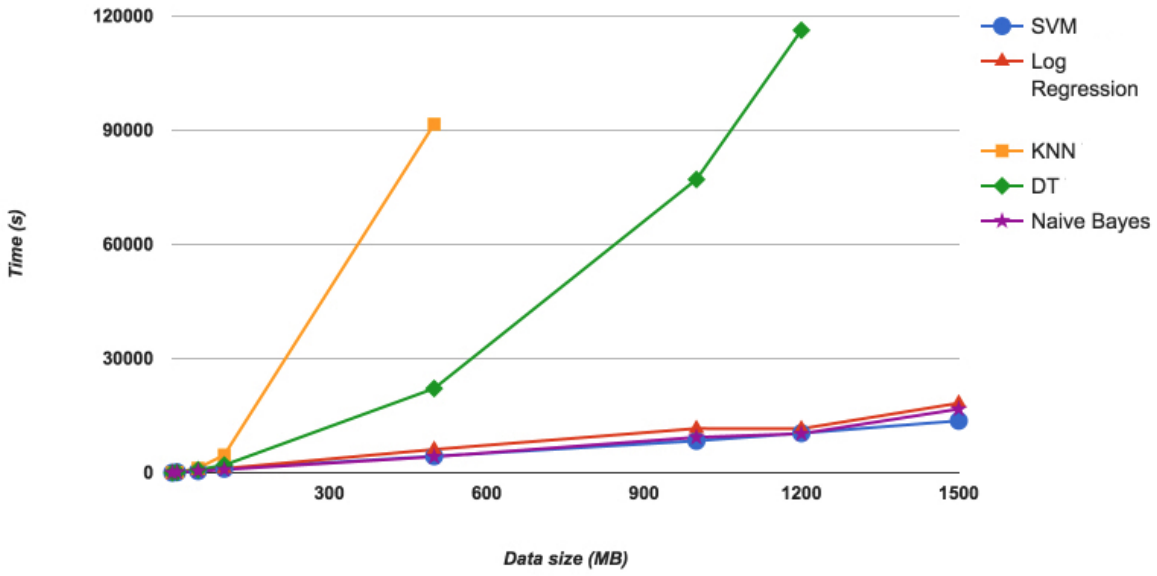

Figure 6.7: Time efficiency vs. Data size.

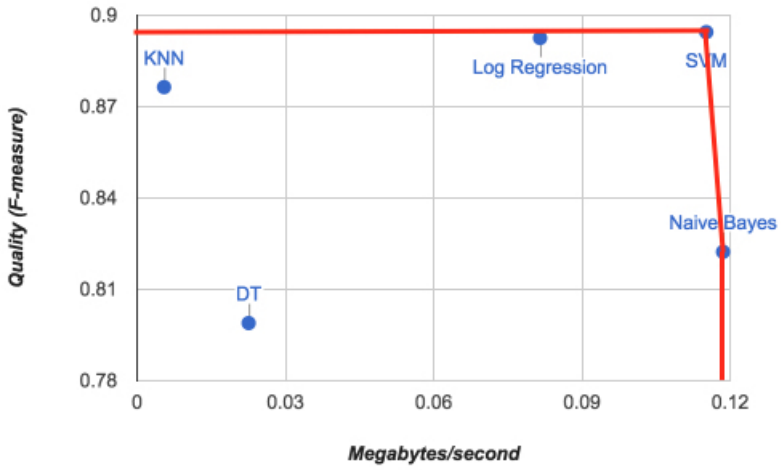

Figure 6.8: Dominant algorithms on news classification for 500 MB data size.

6.3.3.3 Dominant Algorithms

Figures 6.8, 6.9, and 6.10 show one of best dominant algorithm that we achieved to show a trade-off between quality and Megabytes/seconds. SVM and Naïve Bayes are in the dominant algorithm in data size less than 1.3 GB. Logistic regression, KNN and DT are not efficient algorithm in Figures 6.8, 6.9, and 6.10. SVM is a choice that would be selecting high quality, but a bit slower than Naïve Bayes. Naïve Bayes is a choice that would be selected not for high quality, but rather as fast algorithm. The speed difference between SVM and Naïve Bayes is close in large data size around 1.2 GB. SVM shows faster than Naïve Bayes in 1.5 GB data size. We can conclude that SVM is dominant and efficient clearly in all data size.

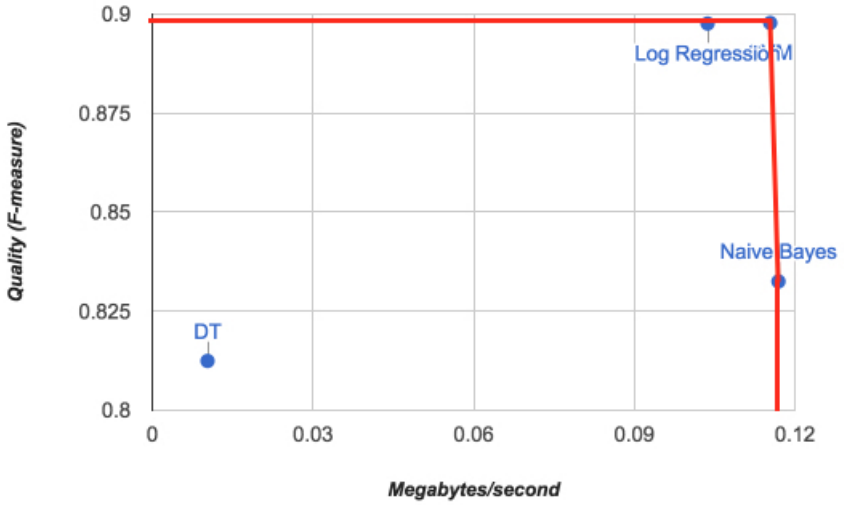

Figure 6.9: Dominant algorithms for news classification for 1.2 GB data size.

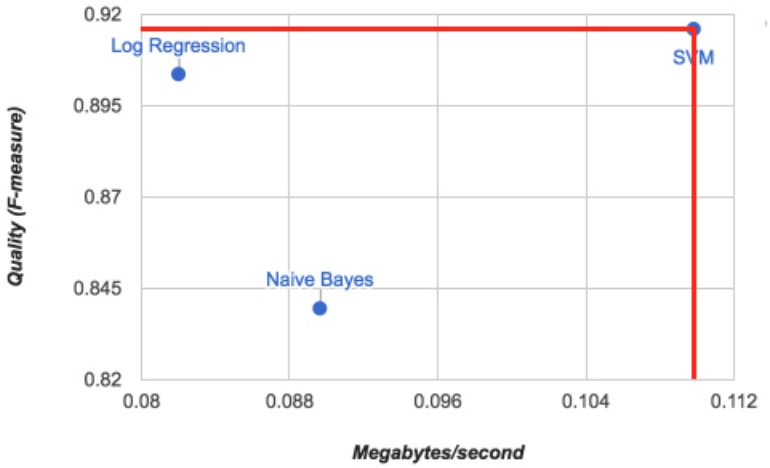

Figure 6.10: Dominant algorithms for news classification for 1.5 GB data size.

6.3.3.4 Overall Comparison

We have two main factors for comparison, quality and time, both are measured per classifier on different size of data as shown in the Figure 6.11. We use the formula (3.4) an equation that can compare between quality, time and size. Quality is scaled by the size of the data and penalized by the time consumed. This way, High quality on a large dataset and low time will have very high quality. Likewise, high quality on a large dataset with high time will have less quality and of course low quality on a large dataset with low time, will also have less quality. Therefore, we can determine which classifier is performing better on both quality and time scale and how much the size variable is affecting these values.

As expected, overall SVM performed the best, since it has the highest quality and lowest time. Also, Decision Trees and KNN performance decreased as we increased the size of the data due to the huge amount of time used in the training process. The confusing part is the comparison between Logistic Regression and Naïve Bayes. Although Logistic Regression per-

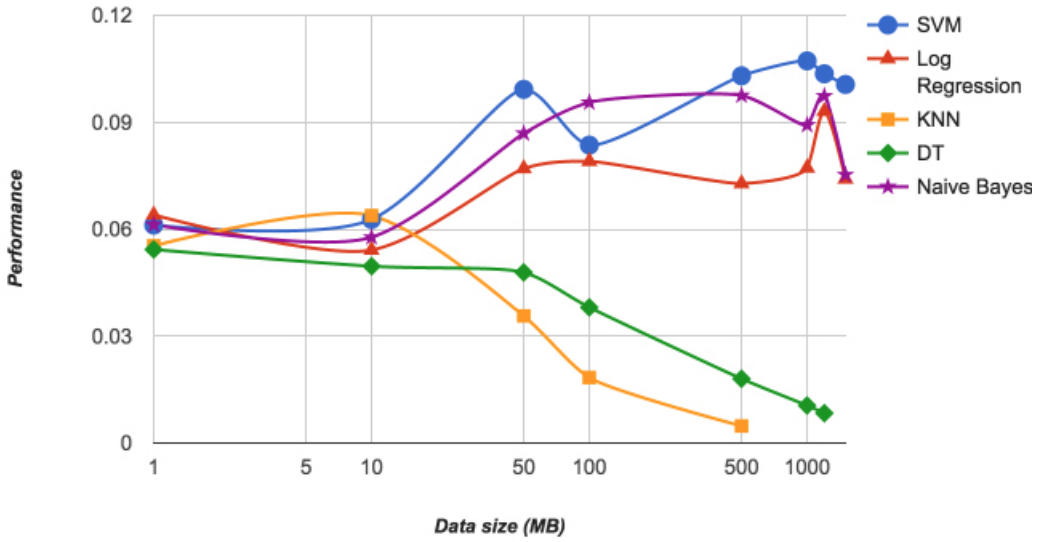

Figure 6.11: Comparing the performance of all algorithms for news classification.

formed better on quality metric, on the proposed metric Naïve Bayes performed better. This is because Logistic regression is better on the quality metric by 7% but worse on the time metric by 12%. So it makes sense that Naïve Bayes performed better than Logistic regression on the proposed metric by 4%.

Manual rules do not require training data, but generally need hand-tuning on the development data. An appealing aspect of decision trees (relative to SVMs and logistic regression) is that they are easily interpreted and modified. Naïve Bayes does require training data, but it is comparatively fast to train.

6.3.4 Analysis

If we want to order the algorithms by quality, they would be: SVM, Logistic regression, KNN, DT, and Naïve Bayes in small data sizes. In the size 1.2 GB, the order would change to SVM, Logistic regression, Naïve Bayes, and DT. By considering time (fast-slow) we would order them as, Naïve Bayes, SVM, Log regression, DT, and KNN in smaller data sizes. This order would be the same without KNN in the larger data sizes as data grows. In this situation, KNN does not work anymore. SVM is the fastest algorithm in 1.5 GB data size and larger.

Logistic regression shows the best performance on 1 MB data sizes, after that KNN in 10 MB and SVM in 50 MB and Naïve Bayes in 100 MB and SVM shows best performance on 1 GB and larger data sizes. Table 6.2 shows the best algorithm in quality, speed, and performance in 1.2 and 1.5 GB data sizes.

SVM has better performance overall in the all data sizes. Naïve Bayes is fastest because of the model parameters of the algorithm. Naïve Bayes uses very basic operations to calculate apriori and conditional probability. Log regression has a bit less quality than SVM in all the data sizes. SVM is faster than Log regression. This involves trivial arithmetic operations, like addition and multiplication and further normalization is only a division by a scalar. KNN is prohibitively expensive for large datasets. It takes a long time to run on a bit big data size. KNN crashed on 500 MB after hours of running.

SVM and Naïve Bayes are dominant algorithms in data sizes less than 1.3 GB. Naïve Bayes

Data size	Quality	Speed	Performance
1.2 GB	SVM	Naïve Bayes	SVM
	Logistic Regression	SVM	Naïve Bayes
	Naïve Bayes	Logistic regression	Logistic regression
	DT	DT	DT
1.5 GB	SVM	SVM	SVM
	Logistic regression	Naïve Bayes	Naïve Bayes
	Naïve Bayes	Logistic regression	Logistic regression

Table 6.2: Algorithms that had the best quality, speed, and performance of all algorithms on 1.2 and 1.5 GB.

is fastest and SVM has the best quality among of the other algorithms. The speed difference between Naïve Bayes and SVM is close in larger sizes (around 1.2 GB). In 1.5 GB data size, SVM shows faster than Naïve Bayes. We can clearly say SVM and Naïve Bayes are dominant in smaller data sizes and SVM is dominant clearly in all data sizes.

6.4 Patents Classification

PAIR (Patent Application Information Retrieval) Bulk Data (PBD) is responsible for the USA Patent and Trademark Office's (USPTO) obligation for advancing the culture of open government as described by the 2013 Executive Order to make open and machine-readable data the new default for government information. PAIR Bulk Data is also responsible for customer requests for this data. PAIR Bulk Data allows customers to recapture and download multiple records of USPTO patents and other patent filing statuses at no cost. PBD contains published documents as well as bibliographic, and patent term extension date tabs in Public PAIR from 1981 to now. There is also some data going back to 1931. Customers can download an entire dataset for all index documents. Currently there are over 9.4 million Records inside PBD. PAIR Bulk Data (PBD) allows a client to browse the USPTO Public PAIR data, requesting one-time bulk download, and formulating queries. This allows client users to interface and build atop a RESTful Application Program Interface (API) supporting full-text and field-specific searches on patent data [USPTO.gov, 2017].

The PBD API helps the developers take advantage of custom search syntax beyond that provided by the client. Innovators and entrepreneurs worldwide are being encouraged by the US Patent and Trademark office to publish their inventions for worldwide use and adoption. For this reason, they have opened the PBD API to some third party developers inside and outside of government. This allows them to benefit the most from the data and allows them to try create their own application [USPTO.gov, 2017].

Up until now, the PBD API has taken advantage of COTS semantics whenever possible. The PBD clients manage an open architecture. The query syntax follows the same standard Apache Solr Search syntax, and the Json documents returned also follow the Solr response formats [USPTO.gov, 2017].

6.4.1 Dataset: USA Patents

The patent claims research dataset includes thorough information on claims from U.S Patents acknowledged between 1976 and 2016 and U.S. patent applications were published between

Data	Search Field	Display field	Example
Application Number	appId	n/a	99999999
Patent Number	patentNumber	patentNumberFacet, patentN	PP999999
Class / Sub- class	appClsSubCls	appCls, appSubCls, appCls	606/304
First Named Inventor	primaryInventor, primaryInve	primaryInventorFacet	"Mahesh Kan- dula , East Godavari Dus- t
Entity Sta- tus	appEntityStatus		UnDiscounted, Small, Micro
Status	appStatus		Patented Case, Pending or Abandoned
Status Date	appStatusDate	appStatusYear	2010-03- 24T04:00:00
Location	appLocation	appLocationFacet	ELECTRONIC, FILE REPOSI- TORY (FR
Issue Date of Patent	patentIssueDate	patentIssueYear	2015-02- 24T05:00:00Z
title	patent Title		
Filing Date	appFilingDate	appFilingYear	2015-02- 16T05:00:00

Figure 6.12: Example of patent metadata.


```
"{\"applicationPublication\":{\"patentPublicationIdentification\":
{\"publicationNumber\":\"US20110173719A1\",\"publicationDate\":
\"2011-07-14\"},\"webURI\":\"http://titan.etc.uspto.gov:9050/netacgi/
nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=/netahtml/PTO/srchnum.
html&r=1&f=G&l=50&s1=20110173719.PGNR.&OS=DN/20110173719&RS=DN/
20110173719\"},\"grantPublication\":{\"patentGrantIdentification\":
{\"patentNumber\":\"8604275\"},\"webURI\":\"http://titan.etc.uspto.gov:
9000/netacgi/nph-Parser?patentnumber=8604275\"}}"
```


Table 6.3: Data structure for published document.

2001 and 2016. This dataset was derived from the Patent Application Publication Full-Text and Patent Grant Full Text files [USPTO.gov,].

The PBD API is designed to be leveraged for many purposes. At the beginning, it was designed to facilitate the download of bulk bibliographic data. At the end, the RESTful architecture revolved around a query as the core resource [USPTO.gov,]. We used POST/queries several times while refining the desired filters and constraint which will store a set of query filters and constraints and return a preview of the results. Once the correct set of parameters has been identified, we request that the complete set of results be bundled for download using a PUT /queries/queryId/package.[1] [USPTO.gov,].

This link provides us with downloaded patent data that have all the fields in the Patent class by using the API and web scraper to collect all requisite information in JavaScript Object Notation (JSON) format. JSON is an open standard format. It uses human-readable text to transfer data objects consisting of attribute value pairs. It is the main format for API responses and parameters. Figure 6.12 describes some of the mapping from data type to index fields [USPTO.gov, 2017].

The published documents contain structured data in JSON format, including publicationNumber, publicationDate, and links at which the publication and/or grant information may be obtained. Table 6.3 shows this data structure [USPTO.gov, 2017].

The Patent terms also contain JSON format structured data. Table 6.4 shows an example of the data structure patent [USPTO.gov, 2017].

For this reason we use the query below:

data = { "searchText": "*", "qf": "patentTitle" , "fq":["appFilingDate:[2015-1-1T00:00:00Z TO *]"]}

[1]The API is available at https://pairbulkdata.uspto.gov/#/apidocumentation

```
{patentTermJson":"{\"patentTermExtensionData\":
{\"filingDate\":\"2010-05-12\",\"adjustmentTotalQuantity\":
0,\"ipOfficeDayDelayQuantity\":212,\"applicantDayDelayQuantity\":
188,\"extensionTotalQuantity\":24,\"patentTermExtensionHistoryData\":
[{\"recordedDate\":\"2014-03-09\",\"caseActionDescriptionText\":
\"AdjustmentofPTECalculationbyPTO\",\"ipOfficeDayDelayQuantity\":
212},{\"recordedDate\":\"2014-03-09\",\"caseActionDescriptionText\":
\"AdjustmentofPTECalculationbyPTO\",\"applicantDayDelayQuantity\":
188}]}}}
```


Table 6.4: Data structure for patent.

We wrote a program which would search the US patent database for the patents filed in the past twenty years. For each patent, the program can get basic data about the patent and a link to a website containing the patent abstract. By parsing the existing webURI data for each patent, we extract an abstract of the patent and write the result to an output file in JSON format. This file gives a list of patents along with the patent abstracts.

The model class for an output patent format is as follow:

```
class Patent {
private String id;
private String title;
private String status;
private Date filingDate;
private String[] inventors;
private String abstract;
}
```

At the end, we write a program to convert the output file to CSV format and we extract the information above to make our dataset.

6.4.2 Experimental Setup

6.4.2.1 Target Variables and Target Preparation

The goal of this experiment is to generate some training data that can help us to predict the answer to the question " Will the patent be accepted or rejected?" by using the patent information that we extracted on the website. For this reason, we labeled the status as binary granted versus not granted or classified the patents to accepted and rejected. The patent's status contains values below. Due to Table 6.5 we represent the status feature in only two classes as Accepted or Rejected. A reject status has a score 0 and accept status has score 1.

6.4.2.2 Documents Representation

We used IPA and query requests to download data. The data that we collected was about 7.18 GB in size. Figure 6.13 is an example of the dataset.

Number of records in our dataset: $7,063,399$

Status	Reject or accept (0-1)
Notice of Allowance Mailed – Application Received in Office of Publications	0
Docketed New Case - Ready for Examination	0
Response to Non-Final Office Action Entered and Forwarded to Examiner	0
Publications – Issue Fee Payment Received	1
Non-Final Action Mailed	0
Final Rejection Mailed	0
Awaiting TC Resp., Issue Fee Not Paid	0
Advisory Action Mailed	0
Response after Final Action Forwarded to Examiner	1
Publications – Issue Fee Payment Verified	1
Awaiting TC Resp, Issue Fee Payment Verified	1

Table 6.5: Target value preparation.

abstract	filingDate	id	inventors	patentclassif	status	title
An emergency repair kit for tire punctures is d includes a sealant dispenser and a box havir accommodating therein compressor unit that a bottle of the sealant dispenser via an air inp	2015-11-12T05:00:0	14940011	Wen-San	141	Publications -- Issue Fee Payment Rec	EMERGENCY REPAIR KIT FC
A sealant dispenser, which can cooperate wit repairing punctured tires, is disclosed. The se includes a bottle with a top opening being fille sealant, a cap mounted over the top opening	2015-10-31T04:00:0	14929334	Wen-San	141	Publications -- Issue Fee Payment Rec	SEALANT DISPENSER
A reflective mirror is provided with a base and including first layers and second layers lamin base and capable of reflecting at least a porti The multilayer film is provided with a first por	2015-04-20T04:00:0	14402666	Yoshio	355	Notice of Allowance Mailed -- Applicati	REFLECTOR, PROJECTION C
An aircraft-side aircraft data retrieval system a data storage device (14) located in a first loca (8), in a aircraft (2) adapted to, during a flight during the flight; and wireless apparatus (18)	2015-11-20T05:00:0	14892984	CHRISTO	455	Response to Non-Final Office Action E	DATA RETRIEVAL SYSTEM IN
In an electro-optical device, a mirror that is fo substrate is sealed by a frame shaped space light-transmitting cover which is adhered to th barrier layer is formed on an outer face of the	2016-02-10T05:00:0	15040725	Terunao	359	Response after Final Action Forwardec	ELECTRO-OPTICAL DEVICE,
An electronic component unit includes a hous housing space that houses an electronic com divided into a plurality of divided spaces by a protruding from a bottom of the housing spac	2015-04-21T04:00:0	14692251	Hirotaka	174	Response after Final Action Forwardec	ELECTRONIC COMPONENT U
A connector assembly is provided and include second connector. The first connector include and a first terminal with a first cantilever secu insulation body. The second connector conne	2015-09-08T04:00:0	14847650	Ming	439	Response to Non-Final Office Action E	Connector Assembly

Figure 6.13: Data sample example.

6.4.2.3 Feature Extraction and Feature Selection

Our dataset did not have any null values. All the text was converted to lowercase; therefore all the words are treated the same. All stop words and punctuation signs are removed from the

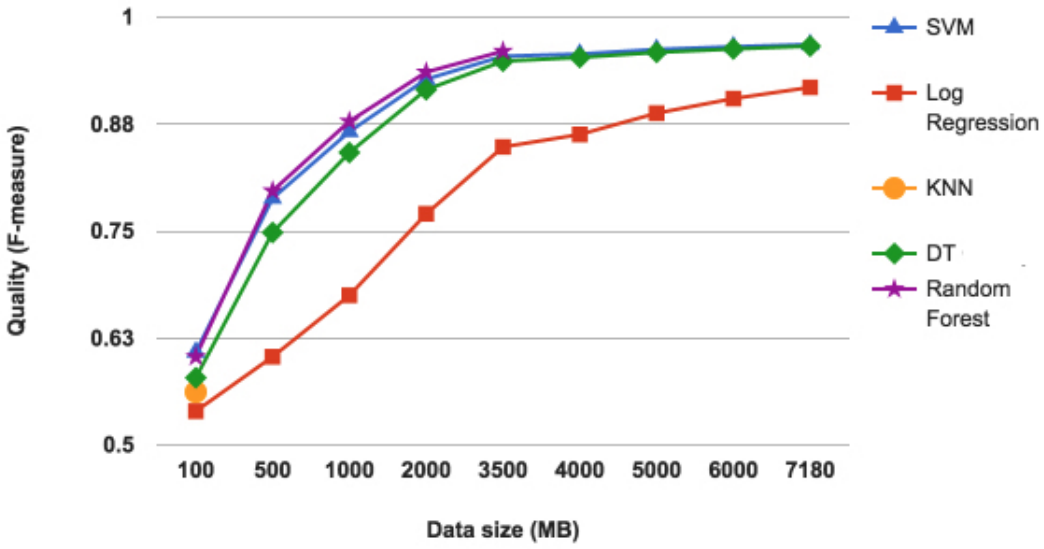

Figure 6.14: Quality vs. Data size.

text. Then tokenizing the text in which the documents are treated as strings and then are partitioned into lists is the last step before feature selection. Term Frequency and Inverse Document Frequency are applied for feature selection.

6.4.3 Experimental Results

As data size is big we used $80:20$ training-testing spit. We used 80% of data for training and 20% of data for testing. To obtain precision, recall and F-measure, as well as confusion matrix was used to calculate TP, TN, FN, FP.

6.4.3.1 Quality Comparison

Figure 6.14 shows the comparison of classification algorithm on various data sizes. As we saw in the experience before KNN crashed after 100 MB data and could not handle working on large data sizes. Random forest and SVM presented good quality, but after increasing data to 3.5 GB Random forest crashed. Logistic regression did not perform with good quality in this kind of dataset.

6.4.3.2 Time Efficiency Comparison

As Figure 6.15 shows SVM and Log regression were the fastest algorithms in this experiment. As experienced before, Random forest was one of the slower algorithms in our experience. DT looks better than Log regression in this example.

6.4.3.3 Dominant Algorithms

SVM is the only algorithm which is dominant in all the charts as shown in Figures 6.16-6.21. Logistic regression did not have as good quality, but it was fast, just not as fast as SVM.

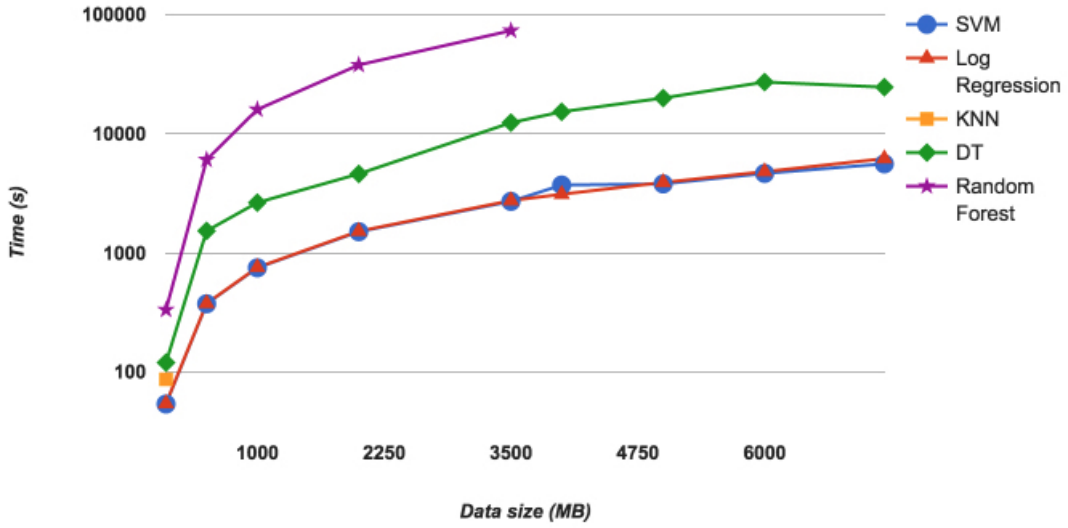

Figure 6.15: Time Efficiency vs. Data size.

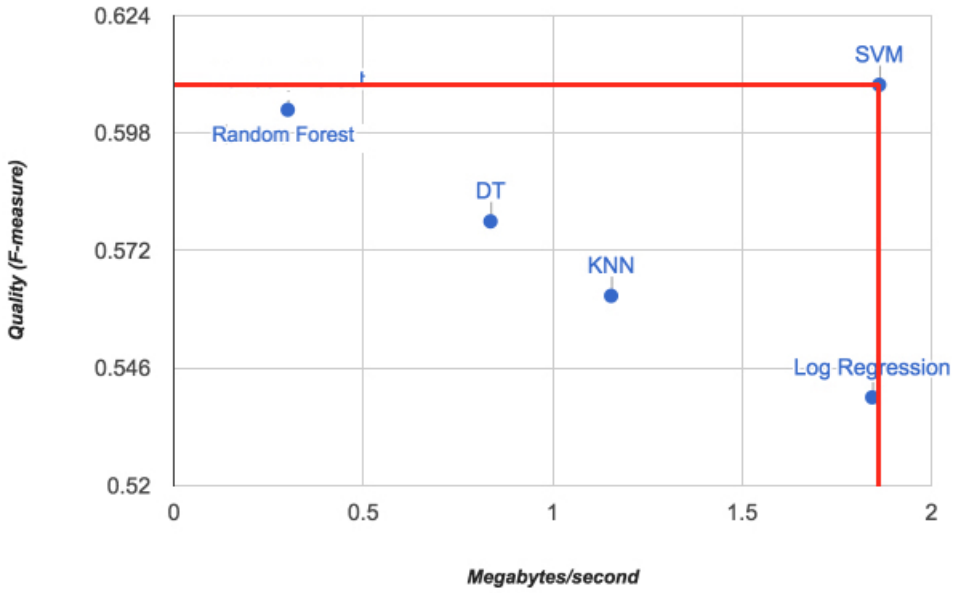

Figure 6.16: Dominant algorithms on patents classification for 100 MB.

By increasing data, quality is increased. In data size 1 GB, Random forest, Logistic Regression and SVM are dominant algorithms as is shown in Figure 6.18. Random forest shows best quality with slow speed and Logistic Regression was fast, but not good quality. SVM shows good quality and good speed.

For 4 GB data size, Random forest crashed. Therefore, SVM showed best quality and Logistic Regression was best in speed. SVM and Logistic Regression are dominant algorithms for 4 GB data sizes.

For 4 GB data sizes, Random forest crashed. Therefore, SVM showed best quality and Logistic Regression was best in speed. SVM and Logistic Regression are dominant algorithm

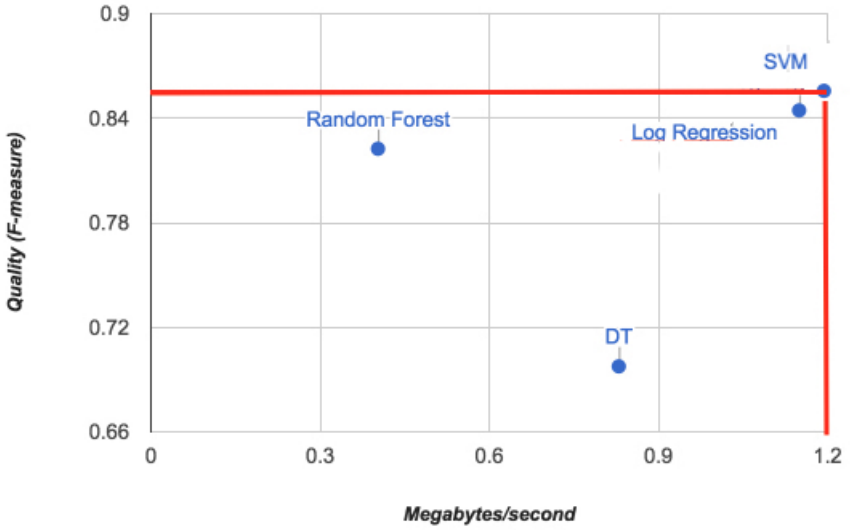

Figure 6.17: Dominant algorithms on patents classification for 500 MB.

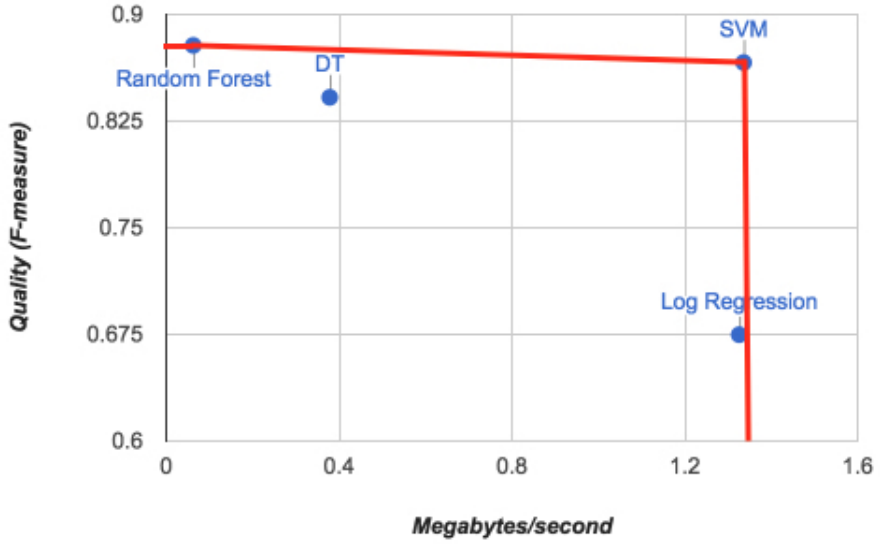

Figure 6.18: Dominant algorithms on patents classification for 1 GB.

in 4 GB data size.

For 6 GB, SVM perform faster than Log regression. SVM in 7 GB still has best quality, and It is the only algorithm that is dominant.

6.4.3.4 Overall Comparison

In the overall comparison, Figure 6.21, SVM shows good performance in large data sizes. In the small sizes of less than 100 MB, Logistic Regression had the best performance. KNN. In our previous experiments, KNN could not handle large data and most of the time after 100 MB crashed. Logistic Regression shows good performance. It did not have good quality, but

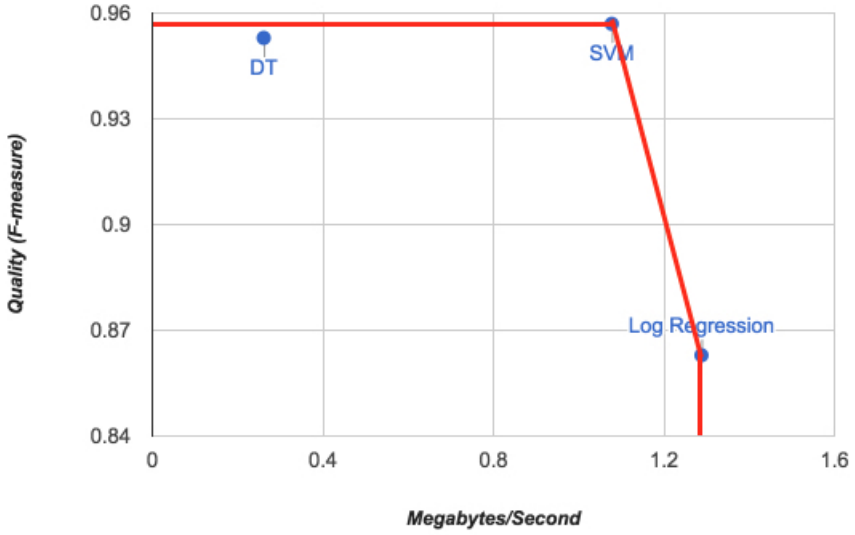

Figure 6.19: Dominant algorithms on patents classification for 4 GB.

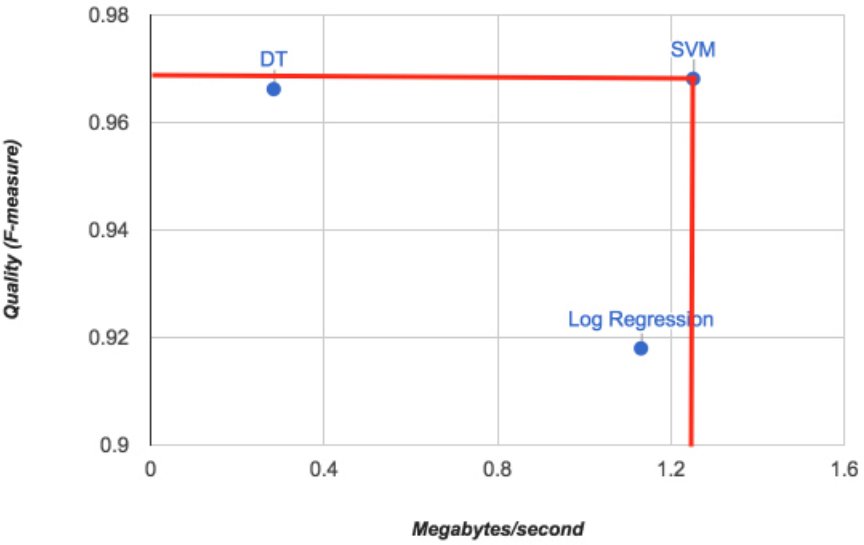

Figure 6.20: Dominant algorithms on patents classification for 7.18 GB.

because it was a fast algorithm, the performance was high. DT had better quality than Logistic Regression, but slower time. Once that data size was over 400 MB, Log regression surged way ahead.

6.4.4 Analysis

For our evaluation, we also used 5-fold cross validation, we almost got the same result as 80:20 training-test. If we order the algorithms by quality, we have: SVM, Random forest, DT, KNN and Logistic regression in smaller data sizes. In the size 500 MB to 3.5 GB, this order would be Random forest, SVM, DT, and Logistic regression. Logistic regression was worst in quality in patent classification. By only considering a running time (fast-slow) the order would be:

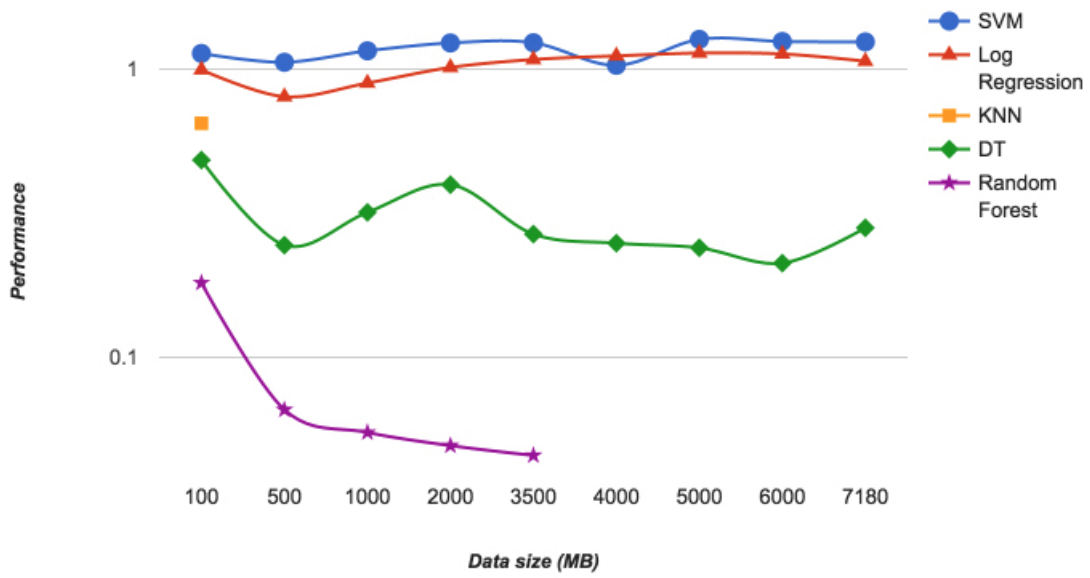

Figure 6.21: Comparing the performance of all algorithms for patents classification.

Data size	Quality	Speed	Performance
7.18 GB	SVM	SVM	SVM
	DT	Logistic Regression	Logistic Regression
	Logistic regression	DT	DT

Table 6.6: Algorithms that had the best quality, speed, and performance on 7.18 GB data size.

SVM, Logistic regression, KNN, DT, and Random forest in all data sizes. KNN crashed after ten hours on 100 MB. Random forest shows better quality than SVM in larger data sizes, but it is so much slower than SVM. For example, SVM is 27 times faster than Random forest in the 3.5 GB data size.

SVM has better performance overall in the all data sizes. And after SVM, Logistic regression, DT, and Random forest show better performance. Random forest has good quality, but because of slow speed it stands in last place for performance. Random forest crashed after several hours the 3.5 GB data size. The best algorithm for quality, speed, and performance in these experiments is shown in Table 6.6.

Another experiment that can be done is using our extracted data set, and trying to classify patents by their content. Each patent belongs to a class and subclass as shown in Figure 6.13. By using the abstract of a patent, we can find to which class the patent belongs.

6.5 Discussion

This chapter is divided into two different document classification problems on different datasets (news classification and patents classification). In both cases, quality increased by adding more data. All the algorithms had the same learning curves in quality. But the shape of performance curves was different. In addition, algorithms showed different behavior on different data sizes. The result shows again that adding more data does not always helps to improve the

performance.

In this chapter we also did a K-fold evaluation for both cases, books and patents. However the results did not changed and that is why were not included in detail.

SVM and Logistic regression in news classification and Random forest and SVM in patents classification are the best algorithms. Random forest and DT show good quality because of the kind of data structure the patent dataset has.

SVM and Naïve Bayes are dominant algorithms in smaller data sizes. SVM is dominant in large data sizes. Random forest, KNN, and DT in order (slow-fast) are a lot more time consuming algorithms. SVM, Naïve and Logistic regression are the fastest algorithms for text classification problems.

SVM shows the best performance in larger datasets. Logistic regression and SVM have close performances. Logistic regression was twice as fast as DT and six times as fast as Random forest in 100 MB. With increasing data, time will increase. Now if we use the same time frame for both algorithms, DT and Logistic regression, or Logistic regression and Random forest, the quality of Logistic regression will increase more than the quality obtained from other algorithms.

KNN did not work in 500 MB and larger. Random forest did not work on larger than 3.5 GB data size.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Now that there is so much availability of digital documents on the World Wide Web and the rapid growth of the volume of data, the task of extracting information from documents in an automatic way, also known as knowledge discovery, has become very important and challenging. The key method for extracting information or knowledge discovery is using a combination of ML, NLP, and data mining. There are some problems which are solved with different methods and a lot of research has been done on different problems/tasks using small datasets to try to improve quality and comparing methods on problems. However, not enough research exists that compares the quality of these methods when considering quality and running time on various data sizes. With the ever growing volume of data with increasingly large data sizes, research will have to consider testing and comparing methods on larger datasets.

We selected three problems in text processing that are usually solved with supervised machine learning. We compared the performance of several methods based on the quality of the results returned, the running time of the algorithms, and the size of the dataset used. We discussed the trade-offs between quality and time efficiency, defining a simple performance measure as quality multiplied by data size and divided by running time. In this way, we can compare more fairly different algorithms. We also find the dominant algorithms on the each of the problems in different data sizes.

The problems considered here included Named Entity Recognition, Sentiment Analysis, and Document Classification. Depending on the problem, kind of data, and number of samples as well as features, the algorithms exhibited different behaviors. For example, SVM had the best performance in larger data sizes. Naïve Bayes and KNN showed better performance on smaller data sizes rather than larger sizes. However, KNN could not work on large data.

Through our work, we found qualitative differences between performance of methods on smaller data sizes and larger data sizes. We described the effect that dataset size has on a classifier accuracy. We examined how algorithms have different behaviors when data size is increased, and noted that the high dimensionality of the features extracted from text documents makes classification of text documents a difficult task.

Table 7.1 shows the summary of three problems, datasets, data size, number of features, evaluation metrics used on each problem, after the cleaning and normalization process.

In conclusion, depending on the interest in quality or speed, we would recommend the use of SVM or Naïve Bayes for text classification or categorization when the data size is small. Naïve Bayes requires only a small amount of training data to estimate the constraints parameters necessary for classification. If the number of samples is huge, we would recommend the

Dataset	Object	Size	Folds	# Features
News	806,791 documents	1.2 GB	6-Folds	8
Movies and TV	1,756,268 reviews	1.50 GB	Training/test	10
Book	2,615,352 reviews	3.35 GB	Training/test & 10-Folds	10
Patents	7,063,399 documents	7.18 GB	Training/test & 6-Folds	6

Table 7.1: Summary of problems, final datasets, evaluation metrics, and features.

Problem	Dominant Algorithm	Best Performance
Named Entity Recognition	SNER (CRF)	SNER (CRF)
Semantic Analysis	SVM	SVM
Document Classification	SVM	SVM

Table 7.2: Dominant algorithm and best performance for all problems in the largest data size.

use of SVM. Table 7.2 shows that SVM was the only algorithm that was dominant and had the best performance in the largest data size in both experiments. If the data has many duplicates, DT and Random forest might be a reasonable choice, as it worked well in datasets which included a significant amount of duplicate data. However, these methods are time consuming. It is worth noting, however, that even here they were outperformed by SVM.

7.2 Future Work

The first immediate way to extend our work is to do more experiments to cover better the parameter space of the problem of comparing ML algorithms. That implies try more datasets where the notion of dominant algorithm can be extended [Tax et al., 2015], as well as trying all possible evaluation techniques. Another extension would be to vary the number of features and consider more algorithms.

As we explained in this thesis, very large data sources tend to contain a lot of detail in the data. Therefore, it is necessary to extract information, process, and organize this large data. One of the problems we face is finding large size datasets. Now it is time to make new datasets by gathering some large data that can be used for research instead of using the available small datasets over and over. This research can continue on different kinds of data with semi-supervised and unsupervised learning methods.

There is a new view that methods can be improved in performance by comparing them with large datasets. Everybody can submit a new method and compare it with previous methods on different datasets. Therefore, we should find a way to verify the performance by considering the quality, running time, and data size. Other work that could be continued would be finding the threshold point where performance no longer improves by adding more data. To find the best performance levels, we must be able to estimate the size of the annotated sample required to reach the best performance. In order to generate efficient models, supervised learning methods need annotated data. When dealing with huge datasets we must know what machine learning algorithm limitations we face. There is a need to understand the limitations of machine learning algorithms at scale while dealing with massive datasets.

Additional questions are: Why does the performance change with adding more data? Can we obtain better quality by adding more features? How do the different methods behave with increasing noise, sparsity and redundancy of data? All of the above subjects, open new interesting trade-off challenges in algorithm analysis and design for NLP and ML. SVM shows best

result in overall classification, can we use SVM in NER?

Finally, more work can be done using the dataset that we extracted to classify patents by their content. Each patent belongs to a class and subclass. By using the abstract of a patent, we can find to which class patents belong. Having patents classified in different ways might be invaluable to people trying to start new ventures.

Bibliography

[Asahara and Matsumoto, 2003] Asahara, M. and Matsumoto, Y. (2003). Japanese named entity extraction with redundant morphological analysis. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 8–15, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Asch, 2013] Asch, V. V. (2013). Macro-and micro-averaged evaluation measures.

[Atdağ and Labatut, 2013] Atdağ, S. and Labatut, V. (2013). A comparison of named entity recognition tools applied to biographical texts. In *Systems and Computer Science (ICSCS), 2013 2nd International Conference on*, pages 228–233. IEEE.

[Ausiello et al., 2012] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (2012). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer Berlin Heidelberg.

[Baeza-Yates, 2013] Baeza-Yates, R. (2013). Big data or right data? In *Alberto Mendelzon Workshop 2013*.

[Baeza-Yates and Ribeiro-Neto, 2011] Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern information retrieval: The concepts and technology behind search*. Addison-Wesley, Pearson.

[Bakshi, 2012] Bakshi, K. (2012). Considerations for big data: Architecture and approach. In *2012 IEEE Aerospace Conference*, pages 1–7.

[Banko and Brill, 2001] Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 26–33, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Bhalla, 2016] Bhalla, D. (2016). `http://www.listendata.com/2014/11/random-forest-with-r.html`.

[Bikel et al., 1997] Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 194–201, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Borthwick et al., 1998] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. (1998). Nyu: Description of the mene named entity system as used in muc-7. In *In Proceedings of the Seventh Message Understanding Conference (MUC-7*.

[Breiman and Cutler, 2016] Breiman, L. and Cutler, A. (2016). https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

[Brownlee, 2016] Brownlee, J. (2016). Python machine learning. http://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/. 2016-05-25.

[Cachopo, 2007] Cachopo, A. M. d. J. C. (2007). *Improving methods for single-label text categorization*. PhD thesis, Universidade Técnica de Lisboa.

[Carpenter, 2013] Carpenter, B. (2008–2013). Alias-i lingpipe 4.1.0. http://alias-i.com/lingpipe.

[Cer et al., 2010] Cer, D. M., de Marneffe, M.-C., Jurafsky, D., and Manning, C. D. (2010). Parsing to stanford dependencies: Trade-offs between speed and accuracy. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *LREC*. European Language Resources Association.

[Charniak, 2001] Charniak, E. (2001). Unsupervised learning of name structure from coreference data. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Chen et al., 2004] Chen, C., Liaw, A., and Breiman, L. (2004). Using random forest to learn imbalanced data. volume 110.

[Chen and Zhang, 2014] Chen, C. P. and Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314 – 347.

[Cios et al., 2007] Cios, K., Pedrycz, W., Swiniarski, R., and Kurgan, L. (2007). *Data Mining: A Knowledge Discovery Approach*. Springer US.

[Dimitrov, 2005] Dimitrov, M. (2005). A lightweight approach to coreference resolution for named entities in text marin dimitrov, kalina bontcheva, hamish cunningham and diana maynard. Technical report.

[Dudhabaware and Madankar, 2014] Dudhabaware, R. S. and Madankar, M. S. (2014). Review on natural language processing tasks for text documents. In *Computational Intelligence and Computing Research (ICCIC), 2014 IEEE International Conference on*, pages 1–5. IEEE.

[Ekbal et al., 2010] Ekbal, A., Sourjikova, E., Frank, A., and Ponzetto, S. P. (2010). Assessing the challenge of fine-grained named entity recognition and classification. In *Proceedings of the 2010 Named Entities Workshop*, pages 93–101, Uppsala, Sweden. Association for Computational Linguistics.

[Erik and Fien, 2003] Erik, T. K. S. and Fien, D. M. (2003). Availble: http://www.cnts.ua.ac.be/conll2003/ner/000README.

[Fang and Zhan, 2015] Fang, X. and Zhan, J. (2015). Sentiment analysis using product review data. *Journal of Big Data*, 2(1):5.

[Finkel et al., 2005] Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Florian et al., 2003] Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 168–171, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Frakes and Baeza-Yates, 1992] Frakes, W. B. and Baeza-Yates, R., editors (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Gandhi and Prajapati, 2012] Gandhi, V. C. and Prajapati, J. A. (2012). Review on comparison between text classification algorithms. *International Journal of Emerging Trends & Technology in Computer Science*, 1(3).

[Gonçalves et al., 2013] Gonçalves, P., Araújo, M., Benevenuto, F., and Cha, M. (2013). Comparing and combining sentiment analysis methods. In *Proceedings of the First ACM Conference on Online Social Networks*, COSN '13, pages 27–38, New York, NY, USA. ACM.

[Han et al., 2011] Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

[Ikonomakis et al., 2005] Ikonomakis, M., Kotsiantis, S., and Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS Transactions on Computers*, 4(8):966–974.

[Isoni, 2016] Isoni, A. (2016). *Machine Learning for the Web*. Packt Publishing Ltd.

[Jeff Robble, 2008] Jeff Robble, Brian Renzenbrink, D. R. (2008). Multilayer neural networks. `https://www.cs.rit.edu/~rlaz/PatternRecognition/slides/NeuralNetworks.pdf`.

[Ji and Grishman, 2006] Ji, H. and Grishman, R. (2006). Data selection in semi-supervised learning for name tagging. In *Proceedings of the Workshop on Information Extraction Beyond The Document*, pages 48–55. Association for Computational Linguistics.

[Jiang et al., 2012] Jiang, J., Teichert, A., Daumé, III, H., and Eisner, J. (2012). Learned prioritization for trading off accuracy and speed. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, pages 1331–1339, USA. Curran Associates Inc.

[Jurafsky and Martin, 2000] Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[Khan et al., 2010] Khan, A., Baharudin, B., Lee, L. H., Khan, K., and Tronoh, U. T. P. (2010). A review of machine learning algorithms for text-documents classification. In *Journal of Advances In Information Technology, VOL.*

[Khan et al., 2009] Khan, A., Bahurdin, B. B., and Khan, K. (2009). An overview of e-documents classification. In *International Conference on Machine Learning and Computing, Singapur*.

[Kobsa, 2007] Kobsa, A. (2007). The adaptive web. chapter Privacy-enhanced Web Personalization, pages 628–670. Springer-Verlag, Berlin, Heidelberg.

[Kong and Smith, 2014] Kong, L. and Smith, N. A. (2014). An empirical comparison of parsing methods for stanford dependencies. *CoRR*, abs/1404.4314.

[Labatut, 2013] Labatut, V. (2013). Improved named entity recognition through svm-based combination.

[Leskovec, 2013] Leskovec, J. (2013). Amazon product data. `https://snap.stanford.edu/data/web-Amazon.html`.

[Lewis et al., 2004] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397.

[Lewis et al., 2015] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2015). `http://trec.nist.gov/data/reuters/reuters.html`.

[Liaghat, 2016] Liaghat (2016). The effect of corpora size on performance of named entity recognition. *BIDMA 2017 : International Symposium on Big Data Management and Analytics*.

[Liu, 2012] Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

[Ma and Ji, 1999] Ma, S. and Ji, C. (1999). Performance and efficiency: recent advances in supervised learning. *Proceedings of the IEEE*, 87(9):1519–1535.

[Mann and Yarowsky, 2003] Mann, G. S. and Yarowsky, D. (2003). Unsupervised personal name disambiguation. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Mansouri et al., 2008] Mansouri, A., Affendey, L. S., and Mamat, A. (2008). Named entity recognition approaches.

[Mayer-Schönberger and Cukier, 2013] Mayer-Schönberger, V. and Cukier, K. (2013). *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. An Eamon Dolan book. Houghton Mifflin Harcourt.

[McCallum and Li, 2003a] McCallum, A. and Li, W. (2003a). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

[McCallum and Li, 2003b] McCallum, A. and Li, W. (2003b). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Moens, 2006] Moens, M. (2006). *Information Extraction: Algorithms and Prospects in a Retrieval Context*. The Information Retrieval Series. Springer Netherlands.

[Nadeau and Sekine, 2007] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

[Nguyen et al., 2016] Nguyen, N.-T., Iliadis, L., Manolopoulos, Y., and Trawiński, B. (2016). *Computational Collective Intelligence: 8th International Conference, ICCCI 2016, Halkidiki, Greece, September 28-30, 2016. Proceedings*, volume 9876. Springer.

[Pang et al., 2002] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Pasca et al., 2006] Pasca, M., Lin, D., Bigham, J., Lifchits, A., and Jain, A. (2006). Organizing and searching the world wide web of facts-step one: the one-million fact extraction challenge. In *AAAI*, volume 6, pages 1400–1405.

[Perone, 2011] Perone, C. S. (2011). Machine learning :: Cosine similarity for vector space models. `http://blog.christianperone.com/tag/tf-idf/`. Accessed: 2011-09-21.

[Qu et al., 2010] Qu, L., Ifrim, G., and Weikum, G. (2010). The bag-of-opinions method for review rating prediction from sparse text patterns. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 913–921, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Radev, 1998] Radev, D. R. (1998). Mann:2003-dependent descriptions of entities. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 1072–1078, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ratinov and Roth, 2009] Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147 – 155.

[Sagiroglu and Sinanc, 2013] Sagiroglu, S. and Sinanc, D. (2013). Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 42–47.

[Schmidt and Okt, 2000] Schmidt, A. and Okt, M. (2000). `https://www.teco.edu/~albrecht/neuro/html/node18.html`.

[Sekine, 1998] Sekine, S. (1998). Nyu: Description of the japanese ne system used for met-2. In *Proc. of the Seventh Message Understanding Conference (MUC-7)*.

[Smith, 2002] Smith, D. A. (2002). Detecting and browsing events in unstructured text. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 73–80, New York, NY, USA. ACM.

[Srihari and Li, 1999] Srihari, R. and Li, W. (1999). Information extraction supported question answering. In *In Proceedings of the Eighth Text REtrieval Conference (TREC-8*, pages 185–196.

[Stehman, 1997] Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77 – 89.

[Sun, 2010] Sun, B. (2010). Named entity recognition: Evaluation of existing systems.

[Sutton and McCallum, 2006] Sutton, C. and McCallum, A. (2006). An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128.

[Swan and Allan, 1999] Swan, R. and Allan, J. (1999). Extracting significant time varying features from text. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, pages 38–45, New York, NY, USA. ACM.

[Tarekegn and Munaye, 2016] Tarekegn, G. B. and Munaye, Y. Y. (2016). Big data: Security issues, challenges and future scope. *International Journal of Computer Engineering & Technology (IJCET)*, 7(0976-6367):12–24.

[Tax et al., 2015] Tax, N., Bockting, S., and Hiemstra, D. (2015). A cross-benchmark comparison of 87 learning to rank methods. *Information processing & management*, 51(6):757–772.

[Urbansky et al., 2011] Urbansky, D., Thom, J. A., Schuster, D., and Schill, A. (2011). Training a named entity recognizer on the web. In Bouguettaya, A., Hauswirth, M., and Liu, L., editors, *Web Information System Engineering – WISE 2011: 12th International Conference, Sydney, Australia, October 13-14, 2011. Proceedings*, pages 87–100. Springer Berlin Heidelberg, Berlin, Heidelberg.

[USPTO.gov,] USPTO.gov. research datasets. `https://www.uspto.gov/learning-and-resources/ip-policy/economic-research/research-datasets`. Accessed: 2013-04-24.

[USPTO.gov, 2017] USPTO.gov ((accessed Feb. 2017)). Pair bulk data. `https://pairbulkdata.uspto.gov/#/api-documentation`. Accessed: 2016-11-23.

[Wajeed and Adilakshmi, 2009] Wajeed, M. A. and Adilakshmi, T. (2009). Text classification using machine learning. *Journal of Theoretical and Applied Information Technology*, 7(2):119–123.

[Wallin, 2014] Wallin, A. (2014). Sentiment analysis of Amazoan reviews and perception of product features.

[Wikipedia, 2017a] Wikipedia ((accessed Feb. 2017)a). k-nearest neighbors algorithm. `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`.

[Wikipedia, 2017b] Wikipedia ((accessed Feb. 2017)b). Support vector machine. `https://en.wikipedia.org/wiki/Support_vector_machine`.

[Yu, 2008] Yu, B. (2008). An evaluation of text classification methods for literary study. *Literary and Linguistic Computing*, 23(3):327–343.

[Zhang et al., 2004] Zhang, L., Pan, Y., and Zhang, T. (2004). Focused named entity recognition using machine learning. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 281–288, New York, NY, USA. ACM.