2020

Automated Change Detection in Privacy Policies

Andrick Adhikari

Automated Change Detection In Privacy Policies

A Thesis

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Andrick Adhikari

December 2020

Advisor: Rinku Dewri

Author: Andrick Adhikari

Title: Automated Change Detection In Privacy Policies

Advisor: Rinku Dewri

Degree Date: December 2020

Abstract

Privacy policies notify Internet users about the privacy practices of websites, mobile apps, and other products and services. However, users rarely read them and struggle to understand their contents. Also, the entities that provide these policies are sometimes unmotivated to make them comprehensible. Due to the complicated nature of these documents, it gets even harder for users to understand and take note of any changes of interest or concern when these policies are changed or revised.

With recent development of machine learning and natural language processing, tools that can automatically annotate sentences of policies have been developed. These annotations can help a user quickly identify and understand relevant parts of the policy. Similarly a tool can be developed that can help identify changes between different versions of a policy that can be informative for the user. For example, suppose according to the new policy a website will start sharing audio data as well. The proposed tool can help users to be aware of such important changes. This thesis presents a tool that takes two different versions of a privacy policy as input, matches the sentences of one version of a policy to the sentences of another version of the policy based on semantic similarity, and inform the user of key relevant changes between two matched sentences. We discuss different supervised machine learning models that are explored to develop a method to annotate the sentences of privacy policies according to expert-identified categories for organization and analysis of the contents. Different word-embedding and similarity techniques are explored and evaluated to develop a method

to match the sentences of one version of the policy to another version of a policy. The annotation of the sentences are used to increase the efficiency of the matching process. Methods to detect changes between two matched sentences through analysis of the structure of sentences are then implemented. We combined the developed methods for annotation of policies, matching the sentences between two versions of a policy and detecting change between sentences to realize the proposed tool.

The research work not only shows the potential of machine learning and natural language processing as an important tool for privacy engineering but also introduces various techniques that can be utilized for any natural language document.

Acknowledgements

I cannot express enough gratitude to all the people without whom completing this thesis would have been next to impossible.

I want to particularly express my sincere gratitude to Dr. Rinku Dewri for his dedicated supervision and inspiring discussions. His motivational mentoring has been the driving force of my work.

I would also like to thank my defense committee members, Dr. Scott T. Leutenegger and Dr. Young Jin Lee, for reviewing my thesis and for being part of my oral defense committee. Their valuable insights have helped me incredibly in delivering the work.

My heartiest thanks to my parents Reeta and Dwijendra, my sister Neethi for always believing in me.

Last but not the least, I would like to thank my friends for their continuous support and encouragement.

Table of Contents

List of Figures

List of Tables

1. Introduction

A privacy policy is a statement or a legal document (in privacy law) that discloses some or all of the ways a party gathers, uses, discloses, and manages a customer or client's data. Personal information can be anything that can be used to identify an individual, not limited to the person's name, address, date of birth, marital status, contact information, ID issue, and expiry date, financial records, credit information, medical history, where one travels, and intentions to acquire goods and services [22]. A privacy policy explicitly describes whether that information is kept confidential, or is shared with or sold to third parties. ISO/IEC 29100 provides a high-level framework for the protection of user information within an organization [14]. ISO/IEC 29100 establishes 11 privacy principles for competent privacy management in an organization. These principles should be adequately reflected in the respective privacy policy of an organization. The 11 privacy principles are:

- Consent and choice: User consent should be taken to collect and process their data. Users should be lucidly informed of their rights and choices. A policy should explain the implications of granting or withholding consent and provide mechanisms to the users to exercise their choice.
- Purpose legitimacy and specification: An organization should comply with the purpose for data collection, and their privacy policy should clearly communicate that purpose to users with sufficient explanation.
- Collection limitation: The collection of user data should be within bounds of the applicable law and necessity of the stated purpose.
- Data minimization: Contact with user information should be minimized and a "need-to-know" principle should be followed. User information should be deleted periodically.

- Use, retention and disclosure limitation: User information should be used only for the intended purpose and retained only as long as necessary. When the specific purpose expires, the information should be locked if needed to be retained.
- Accuracy and quality: User information should be collected and processed accurately. Information should be verified and the reliability of the information should be ensured.
- Openness, transparency and notice: Organizations should clearly and sufficiently communicate the policies, practices and procedures governing user information. The communication should include purpose of information collection, information disclosure and sharing principles, and retention and disposal principles. User choices and mechanisms to exercise them should be communicated. Users should be notified of any changes.
- Individual participation and access: Users should have the ability to access, review, edit and delete their information in a simple, fast and efficient manner.
- Accountability: All the privacy-related policies, procedures and practices should be documented. Third party accountability should be ensured. A privacy officer should be assigned to enforce accountability.
- Information security: Organizations should protect the confidentiality, integrity and availability of user information. The security of the information should be guaranteed, and compliance with legal requirements and security standards should be ensured. Periodic security risk assessment and a cost/benefit analysis should be conducted. Actions and fail-safes should be implemented for any potential event.
- Privacy compliance: Organizations should ensure their compliance with privacy principles. Periodic privacy audits and internal compliance check should be conducted. A privacy risk assessment process should be developed and maintained.

Various legal regimes around the world require that website operators, app publishers, data processors and service providers post a notice on how they gather and share users' information [40]. This requirement results in a large number of privacy policy documents that most users are unlikely to read due to the incomprehensible

nature of the documents. As a matter of fact, if users start reading policies for each of the services they use, it is estimated that it would cost them at least 181 hours per year [23].

Whenever there is a change in the data collection and use practices followed by a company, the respective policy is revised and modified to reflect that change. It becomes very difficult for users to be informed about the introduced changes. To understand the changes, users will have to read, understand and compare each sentence of the two versions of the policy, which is a very tedious and complicated task. Hence, most users remain in the dark whenever a change in the policy is introduced and are unable to make informed decisions related to their privacy.

1.1 Problem Statement

One of the privacy principles is "openness, transparency and notice". Organizations or companies use privacy policies as the primary document to inform users about their user data governing policies, procedures and practices; any change in them is met with a change in the privacy policy, producing a different version of the document. A different version of the policy may introduce changes that can play a vital role in a user's participation. Suppose, a company decides to start collecting more sensitive information about the user. This information will be provided through a change in the privacy policy. Even if a user is well-versed in that company's privacy policy (unlikely), it will still be an extremely difficult task for the user to be aware of the critical change through perusal. The sheer number of privacy policies (a policy for each of the used websites or services) further demotivates users to be well informed about any changes in those policies.

The goal of this research is thus to work towards a tool that facilitates the comparison of two versions of a privacy policy with ease, and help users learn about important changes between the two documents. Using natural language processing and machine learning, our tool extracts and presents the changes between two versions of a policy in a comprehensible and useful format. The tool takes two different versions of a privacy policy as input, match the sentences of one version of a policy to the sentences of another version of the policy based on semantic similarity, and inform the user of key relevant changes (addition and deletion) between two matched sentences. Figure 1.1 presents workflow to perform automated change detection in privacy policies. Our

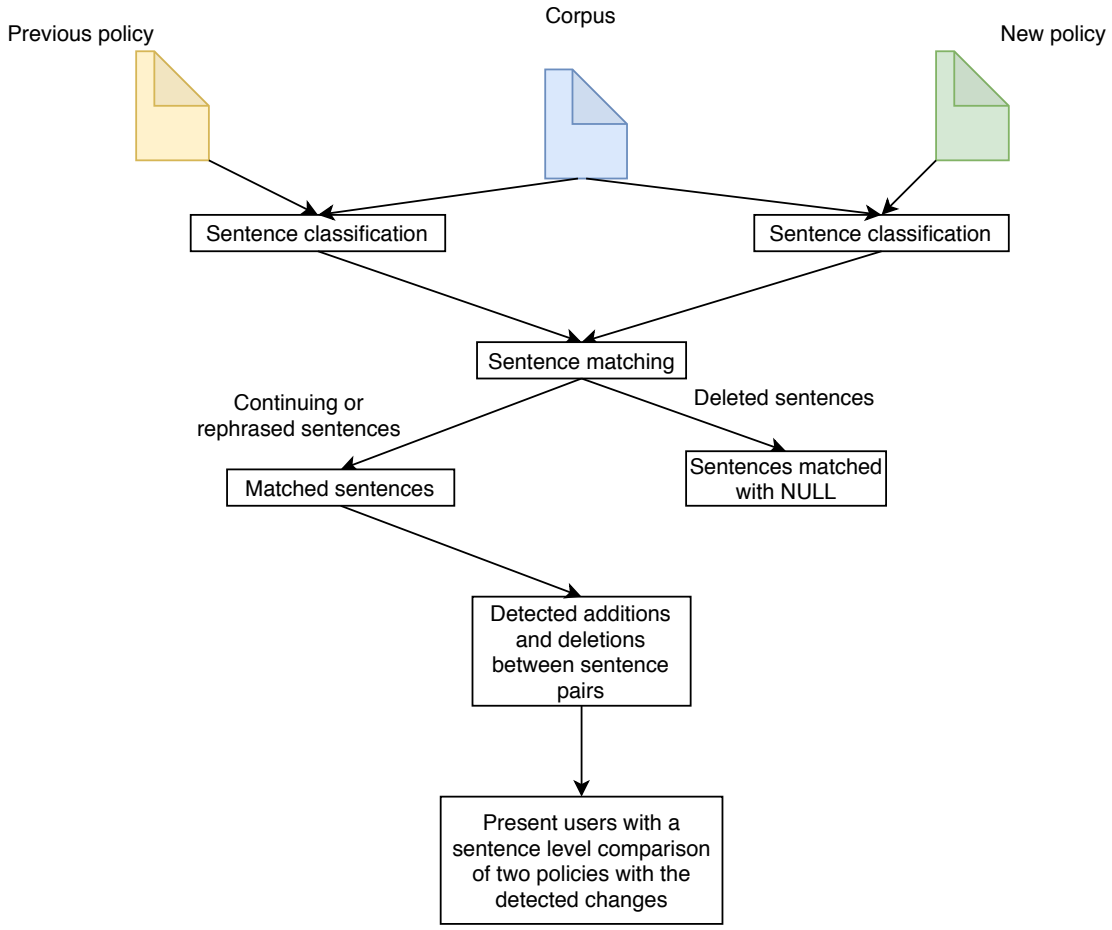

Figure 1.1: Flow chart of tasks for automated change detection in privacy policies

research is divided into three tasks:

- Sentence classification: This task uses expert-identified categories of privacy policies and machine learning to develop tools for classifying sentences in a privacy policy. The categorized sentences helps in understanding, organizing and analyzing the contents of a policy.

- Sentence matching: This task maps sentences from one version of a policy to the next version of the policy. Different permutation and combination of word-embedding techniques and similarity methods methods are implemented, tested and evaluated.

- Change detection: This task involves studying the relationship between words in sentences to devise techniques that can detect addition and deletion changes

between two sentences.

1.2 Evaluation Overview

We shall evaluate a number of methods for sentence classification, sentence matching and change detection in an attempt to identify suitable candidates for the three tasks. Our sentence classification experiments shall use a corpus of 115 privacy policies that have been manually annotated for fine-grained data practices and categories. Primary evaluation for sentence matching and change detection methods shall be conducted using different revisions of the Facebook, WhatsApp and Twitter privacy policies during the period of 2005 to 2019. In addition, we shall extend the evaluation to a total of 79 policy pairs from Facebook, WhatsApp, Twitter, Google, LinkedIn and Snapchat using the final sequence of chosen algorithms.

1.3 Thesis Contribution

The following summarizes the contributions we make through this thesis.

- We bring together a number of existing methods in machine learning and natural language processing, and prior work in usable privacy policies, and apply them to a problem in informed privacy decision making, namely that of automatically detecting privacy-related changes in an organization's privacy policy;
- We present a workflow to perform change detection in privacy policies through a sequence of three filtering tasks (sentence classification, sentence matching and change detection) in order to prune redundant comparisons;
- We present extensive results on the application of our methodology on 79 policies taken from six global technology organizations, and bring to light some observations prevalent in these policies over time; and
- We make a first attempt at creating a web application to visualize privacy policy changes.

1.4 Organization of Thesis

This thesis is structured into the following chapters:

- Related Work and Background: The chapter explores existing research and relevant contributions on change detection, and making privacy policies more usable for users. It describes basic theoretical foundations. Machine learning, natural language processing techniques and data science concepts are discussed.
- Data Description: This chapter discusses the description, collection and use of the data sets used in this research.
- Sentence Classifier: The chapter describes the task and process of developing methods for sentence classification. Observations and results from the task are discussed.
- Sentence Matching: The chapter describes the task and process of developing methods for matching sentences from two versions of a policy. Observations and results from the task are discussed.
- Change Detection: The chapter describes the task and process of developing methods for change detection between two sentences. Observations and results from the task are discussed.
- Extended Evaluation: This chapter discusses an extended evaluation and analysis of our developed methods.
- Conclusion and Future Work: This chapter summarizes the results of this research work and discusses related future work.

2. Related Work and Background

Natural language privacy policies are the primary medium used by websites to communicate information addressing user data collection and use, becoming a de facto standard to address expectations of "notice and choice" on the web [6]. These notice and choice are, at minimum, insufficient and, in many cases, undesirable [3]. Notifying users about a system's data practices is supposed to enable users to make informed privacy decisions. Yet, the long and complicated nature of privacy policies, are often ineffective because they are neither usable nor useful, and are therefore ignored by users [35]. McDonald et al. contend that the time required to read privacy policies is equivalent to a form of payment, where users pay with their time to research policies in order to secure their privacy [23]. Even when users attempt to understand the practices described in the policies, there is often a disagreement between the actual meaning and the user's interpretation [31].

There has been a wealth of research on techniques to make these policies more usable and comprehensible for consumers. Vail et al. conducted an empirical study of different ways of presenting information from privacy policies to consumers, and discuss some of the trade-offs associated with these different types of presentation [36]. Micheti et al. aim to develop guidelines for constructing privacy policies, with the end goal of making them so simple that, even children and teens can accurately interpret them with relative ease [24]. The Platform for Privacy Preferences (P3P) lets websites convey their privacy policies in a computer-readable format, which can then be used by browser agents to automatically check the alignment of the policies with a user's specified privacy preferences [7].

The Usable Privacy Project took the initiative to leverage natural language processing, privacy preference modeling, crowdsourcing, formal methods, and privacy interfaces to develop a practical framework that empowers users to more meaningfully control their privacy, without requiring additional cooperation from website opera-

tors [32]. Under the Usable Privacy Policy Project, there are extensive research works on using natural language processing to understand the content of privacy policies. Liu et al. presented advances in using supervised learning to automatically annotate paragraphs and sentences in a policy with expert-identified categories of policy contents [18]. Ramanath et al. approached the annotation of privacy policy segments as an alignment problem by using hidden markov models [30]. Ammar et al. utilizes logistic regression to predict the presence or absence of a limited set of practice statements within a privacy policy [2]. Wilson et al. studied the accuracy of crowd sourced privacy policy annotations, the levels of granularity in annotations that are feasible for automatic analysis of privacy policies and formulated guidelines that can enhance crowd worker productivity in annotation tasks [38]. Sathyendra et al. and Habib et al. focuses on extracting user choices in privacy policies, particularly opt-out choices [34] [10]. Sathyendra et al. explores a query answering approach that would enable users to ask questions about specific aspects of privacy notices [33]. A similar work is presented by Harkous et al. describing conversational privacy bots (PriBots) that build on machine learning techniques to automatically annotate the text of privacy policies [11]. Cherivirala et al. presented a website that facilitates users with a visualization tool for mapping text segments onto meaningful categories of data collection and use practices [5]. Zimmeck et al. introduced a scalable system to help predict the compliance of Android apps' with privacy requirements by combining machine learning-based privacy policy analysis with static code analysis of apps [39]. Evans et al. identifies information type hyponymy in policies to help create an information type ontology [8]. Hosseini et al. provides empirical methods that can be used to manually create an information type ontology from policies [12]. Oltramari et al. introduced an ontology of data collection and use practices that allows one to submit SparQL queries against a corpus of annotated privacy policies [27]. Wilson et al. describes the creation of a dataset of privacy policies from websites and an annotation scheme desribing the data collection and use practices of the different sections in a privacy policy [37]. Despite tremendous effort and research on making privacy policies more usable for users, there has been no work in detecting changes between revisions of a privacy policy. Most of the current work is focused on making policies more understandable, through organization of the policy content and extraction of relevant information.

There are some research that address change detection in other domains, but none of them are focused on detecting changes in a natural language document. An auto-

mated change-detection algorithm for HTML documents using changes of data contents with respect to markup structures according to the HTML grammar is presented by Lim et al. [17]. Chawathe et al. focuses detecting meaningful changes in hierarchically structured data, such as nested-object data and describing the changes in a semantically more meaningful way [4]. Lim et al. and Chawathe et al. use structural rules for the data to identify changes, and do not use any learning model [17] [4]. There are plenty of research in detecting changes in images and using the identified changes to study a particular area. For example, Mas conceived a large range of methodologies for identifying environmental changes using change detection in satellite images [21]. Lienhart discusses various methods for detecting video shot boundary using change detection in pixel data [16].

We conclude that there has been a lack of research using natural language processing and modern machine learning techniques to detect changes in natural language documents. This implies that the potential for natural language processing in change detection is still untapped. Through this research, we not only hope to make privacy policies more "user friendly", but also show another use case of NLP that can be potentially applied in other domains as well.

Due to the large versatility and enormous number of different applications in NLP and machine learning, the chapter will only focus on subjects, which are important to this thesis.

2.1 Natural Language Processing

Natural language processing or NLP is a subfield of artificial intelligence (AI) concerned with computer and human (natural language) interactions. NLP combined with privacy engineering is the main area of research in this thesis. This section serves to put natural language processing (NLP) in the context of artificial intelligence (AI) and discusses components of natural language processing.

2.1.1 NLP in AI

Artificial intelligence is a very broad term generally used to refer to systems that can in a way think [13]. For this thesis, the interpretation of Hurwitz et al. will be taken as it defines NLP in relation to AI. AI consists of four main parts, which are machine learning (ML), reasoning, planning and natural language processing (NLP), as shown

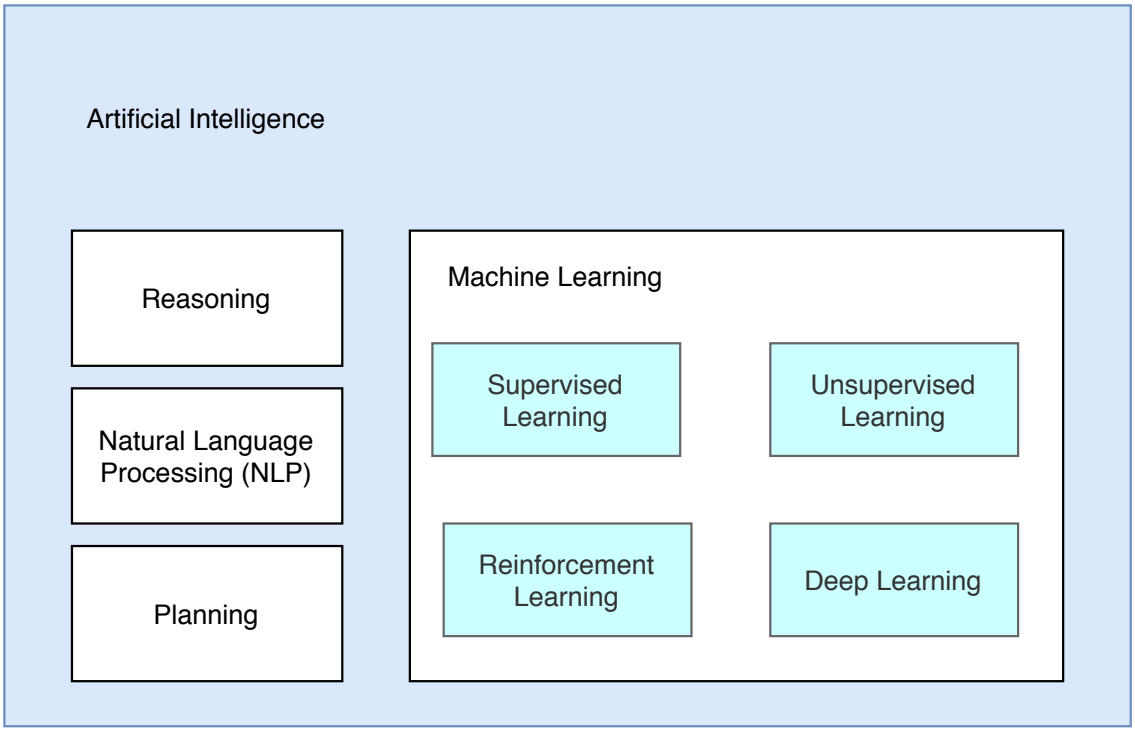

Figure 2.1: Artificial Intelligence contains NLP and Machine Learning

in Figure 2.1. Reasoning enables a machine to provide suggestions based on data, whereas planing empowers systems to act autonomously on the interpretation of data. NLP deals with human and machine interaction using unstructured natural language, as applied to machine translation, speech recognition, dialog systems, named entity recognition, information retrieval and text classification.

2.1.2 Components of NLP

The processing of human language is based on understanding the intended meaning of a message. To correctly interpret a message the machine need to take into account all components of a message such as phonetics, morphology, syntax, semantics and pragmatics. Phonetics is about the acoustic properties of a sound produced by the human vocal tract. It examines how sounds are physically constructed. The sound of a particular human language is studied in phonology. Phonetics and phonology are important in speech recognition applications where sounds are converted to words by computers.

Morphology explores the actual root of the word and concerns the meaning and

architecture of the word. For example the word “flying” is rooted at word “fly”. NLP concepts like stemming and lemmatizing are based on this component.

Syntax refers to ordering and forming of grammatically correct sentences from the words. Semantics on the other hand targets the overall intended meaning of the sentences constructed. Pragmatics is used to get the overall context of the situation where a sentence is used. All the components of natural language are needed to be taken into account for a machine to use it. The following section describes feature selection and preprocessing based on these components.

2.2 Feature Selection and Preprocessing

Feature selection is the process where features that contribute most to the desired output are automatically or manually selected. Preprocessing is the step in which the data gets transformed or encoded so that it can be easily interpreted by the machine. Feature selection and preprocessing are significant tasks in artificial intelligence representing data preparation. Due to unstructured and arbitary nature of text data, this task has significant impact on text analysis [1].

2.2.1 Tokenization

While processing text in NLP, it needs to be broken into smaller units called tokens. Tokenization is used to create these tokens which can be simple words or the whole sentence itself, and are the smallest independent units of natural languages. A simple tokenizer can be realized that splits words based on occurrences of space but may lose useful information like words belonging together semantically e.g bi-grams. Other ways of tokenization include using regular expressions and language specific rules. Tokenization is generally the first step in natural language processing.

2.2.2 Stop Word Removal

Most natural languages have words that occur more than others but do not add much information to the context of the text. In English “the”, “a” , “that” etc. are some examples of such words. These words are referred to as *stop words* and removing them can decrease the size of the raw input in NLP. Stop word removal can be done by simply

checking against a set of standard stop word list, but this task should be done carefully in order to not remove words like “no” or “not” that can change the meaning of the text.

2.2.3 Stemming

Stemming is the technique of mapping words to their word stems, making the unstructured data more accessible for the machine by reducing the input dimensions, e.g. liking to like or retrieval to retrieve. This helps in representing the text in simpler terms. The first stemming algorithm developed in 1968 was based on deleting longest suffixes and spelling exceptions [20]. There are two errors that can occur while stemming: 1) overstemming and 2) understemming. Overstemming occurs when two words are stemmed from the same root, but are of different stems. Understemming occurs when two words are stemmed from different roots, but are not of different stems.

2.2.4 Lemmatization

Lemmatization is similar to stemming. It maps the words to their dictionary type or intended originating structure. Verbs are transformed to their infinite form, a noun is reconstructed to it’s singular representation, and adverbs or adjectives anticipate their positive format [19], e.g. best to good. The method is based on morphological analysis and is supported by dictionary entries. Lemmatization like stemming also makes the input space smaller.

2.2.5 Parts of Speech tagging

Parts of speech tagging or often referred to as pos-tagging is the process of marking up a word to it’s particular part of speech based on not only it’s definition but also it’s context. For English language, there are nine major parts of speech: noun, verb, article, adjective, pronoun, adverb, preposition, conjunction and interjection. There are many more categories and sub-categories, for example, nouns can be further subcategorized as plural, singular and possessive. Pos-tagging is harder than it seems as some words can represent more than one part of speech at different times. This research uses pos-tagging for lemmatization and also in identifying nouns in detecting changes in matched sentences.

2.3 Vector Representation

Word representations should be machine readable and thus, text is often transformed to numerical representations. These representations may only encode the statistics of the word or may include the word's context as well. Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP), where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually, it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension. It is used in information filtering, information retrieval, indexing and relevancy rankings. Given a set of textual documents, a vocabulary can be built, using which documents can be transformed to vectors.

The next few sections discusses the vectorizarion methods that are used and analyzed in this thesis.

2.4 Tf-idf vectorization

As discussed in section 2.3, text documents are transformed into vectors of real number. Suppose $\{w_{11}, w_{12}, \ldots w_{1j}\}$ represents the first sentence in a document and each dimension corresponds to a separate term or word, w_{ij} is the j^{th} word in the i^{th} sentence. If a word occurs in the sentence, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. Td-idf is one of these methods of computing these weights.

Tf-idf stands for term frequency-inverse document frequency, and td-idf weight is often used for feature extraction in text mining. The weight computation is based on a statistical measure that evaluates how important a word is to a document in a corpus. The measure is proportional to the number of times the word appears in the document but is offset by the frequency of the word in the corpus.

2.4.1 Computation

Td-idf weight is composed of two terms: Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency measure how frequently a term occurs in a document, normalized by dividing it by the length of the document as it is possible

that a term may appear more in a longer document than a shorter one.

For a term t, $TF(t) = \dfrac{\text{Number of times term t appears in document}}{\text{Total number of terms in the document}}$.

Inverse Document Frequency (IDF) measures the importance of a term in the document, given by:
For a term t, $IDF(t) = ln\left(\dfrac{\text{Total number of documents}}{\text{Number of documents with term 't' in it}}\right)$.

The final weight of the term t will be given by: $TF(t) \times IDF(t)$.

In this thesis, tf-idf is used for feature extraction during sentence classification using OPP-115 corpus for term weight calculations. Each policy in the OPP-155 corpus is a document in tf-idf computation. The total number of documents is 115 in OPP-115 corpus.

2.5 Word2vec

Word2vec was created and published in 2013 by Mikolov et al. [25]. Word2vec is a group of related models that are used to produce word embeddings. These are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words.

Word2vec uses a large corpus as input and produces a vector space of several hundred dimensions. Each unique word in the corpus is assigned a corresponding vector in space such that words sharing common context are placed close to each other [25]. This captures syntactic and semantic relationships between the words.

2.5.1 Computation

Word2vec is a method to construct an embedding or transform documents to vectors of real number. It can be obtained using two models: skip-gram and common bag of words (CBOW), both involving a neural network that uses back-propagation to train. Skip gram works well with small amount of data and is found to represent rare words well. On the other hand, CBOW is faster and has better representations for more frequent words.
Consider the following sentences

S_1: *"Have a good time"*

and

S_2: *"Have a great time"*

If we construct an exhaustive vocabulary, say V, then, V = {Have, a, good, great, time}. One-hot encoded vectors for each of the word in V will have a length equal to size of V (=5). One-hot encodings for our example vocabulary will be "Have" = [1,0,0,0,0], a = [0,1,0,0,0], good = [0,0,1,0,0], great = [0,0,0,1,0] and time = [0,0,0,0,1]. These are vectors of zero except for the element at the index representing the corresponding word in the V.

If we visualize these encodings on a five dimensional space, each of the word will occupy one of the dimensions and has no projection along other dimensions. This implies that the words have no similarity among themselves. This means "good" and "great" are completely different from one another, which is not true.
We introduce some dependence of one word on the other words, words that are in context of this word will get a greater share of this dependence. Common bag of words and skip gram models use the context of the words to compute vector representation that preserve the dependence between words.

2.5.2 Common Bag of Words Model

This method takes the context of each word as the input and tries to predict the word corresponding to the context. Figure 2.2 is a simple representation of the architecture with only one word in the context. The input is a one hot encoded vector of size V. The dimension of our hidden layer and output layer will remain the same. In figure 2.2, $[x_1, x_2, \ldots x_v]$ is the one hot encoding of the input context word of dimension $1 \times V$. Matrices $W_{V \times N}$ and $W'_{N \times V}$ are initialized with random small values. $[x_1, x_2, \ldots x_v]$ is multiplied with a matrix of $V \times N$ dimension ($W_{V \times N}$) from input layer to hidden layer, returning a vector of dimension $1 \times N$. The goal is to produce probabilities for the target word in the output layer. For the output vector $[y_1, y_2, \ldots y_v]$, y_i is the probability that the word corresponding to index i occurs, when the given input word is present. Consider S_1, let the input to the neural network be the word "good" ([0,0,1,0,0]), whose target word is "time" ([0,0,0,0,1]). Element y_5 in the output vector $[y_1, y_2, \ldots y_v]$ will

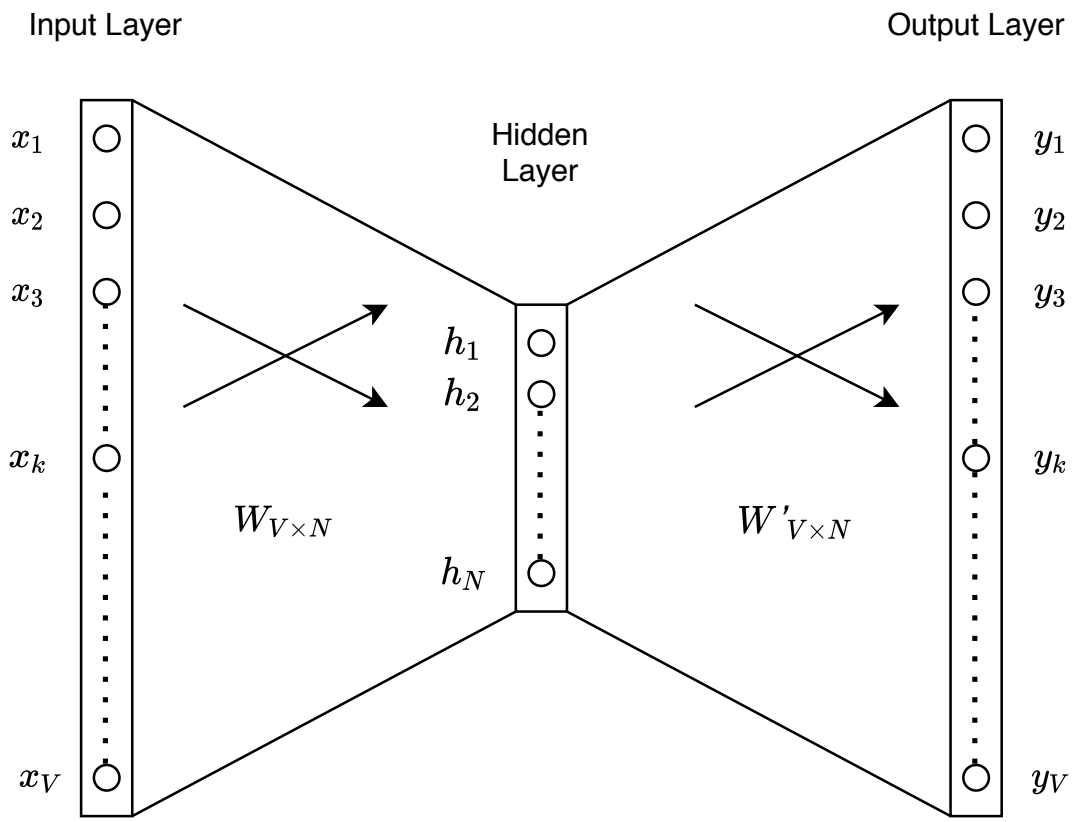

Figure 2.2: Simple CBOW model with only one word in the context

give us the probability P("time" is present | "good" is present). Vector $1 \times N$ from hidden layer is multiplied with a matrix of $N \times V$ dimension $W'_{N\times V}$) and the resultant ($1 \times V$ vector is averaged element-wise to obtain the activation values at the output layer. Activation values of output layer neurons are converted to probabilities using the softmax function, the output of the k^{th} neuron is computed by the following expression:

$$y_k = \frac{exp(activation(k))}{\sum_{n=1}^{V} exp(activation(k))}$$

In the expression, $activation(i)$ represents the activation value of the i^{th} output layer neuron. Given the target vector, the error vector for the output layer is computed by subtracting the output probability vector from the target vector. Once the error is known, the weights in the matrices $W_{V\times N}$ and $W'_{N\times V}$ are updated using backpropagation. In the process, we learn the vector representation of the target word. The neural network is trained by presenting different context-target words pair from the corpus.

This is how CBOW learns relationships between words and in the process develops vector representations for words in the corpus. Figure 2.3 shows the architecture for multiple context words. The input is a $C \times V$ matrix, C is the number of context words for a single target word and input is a $C \times V$ matrix. Each row(x_i) of the matrix($[x_1, x_2, \ldots x_C]$) is the one hot encoding of a context word, where x_i is a vector of size V (size of the vocabulary). Vectors $\{x_1, x_2, \ldots x_C\}$ are averaged together at the hidden layer and the neural network learns the same way it does for a single context word.

2.5.3 Skip Gram Model

Skip gram model is a flipped CBOW model for multiple context words (figure 2.4). The target word is given as input to the neural network that produces a vector of probabilities. The result of the output layer, a $C \times V$ dimensional output is a probability vector of size V for each of the C context words.

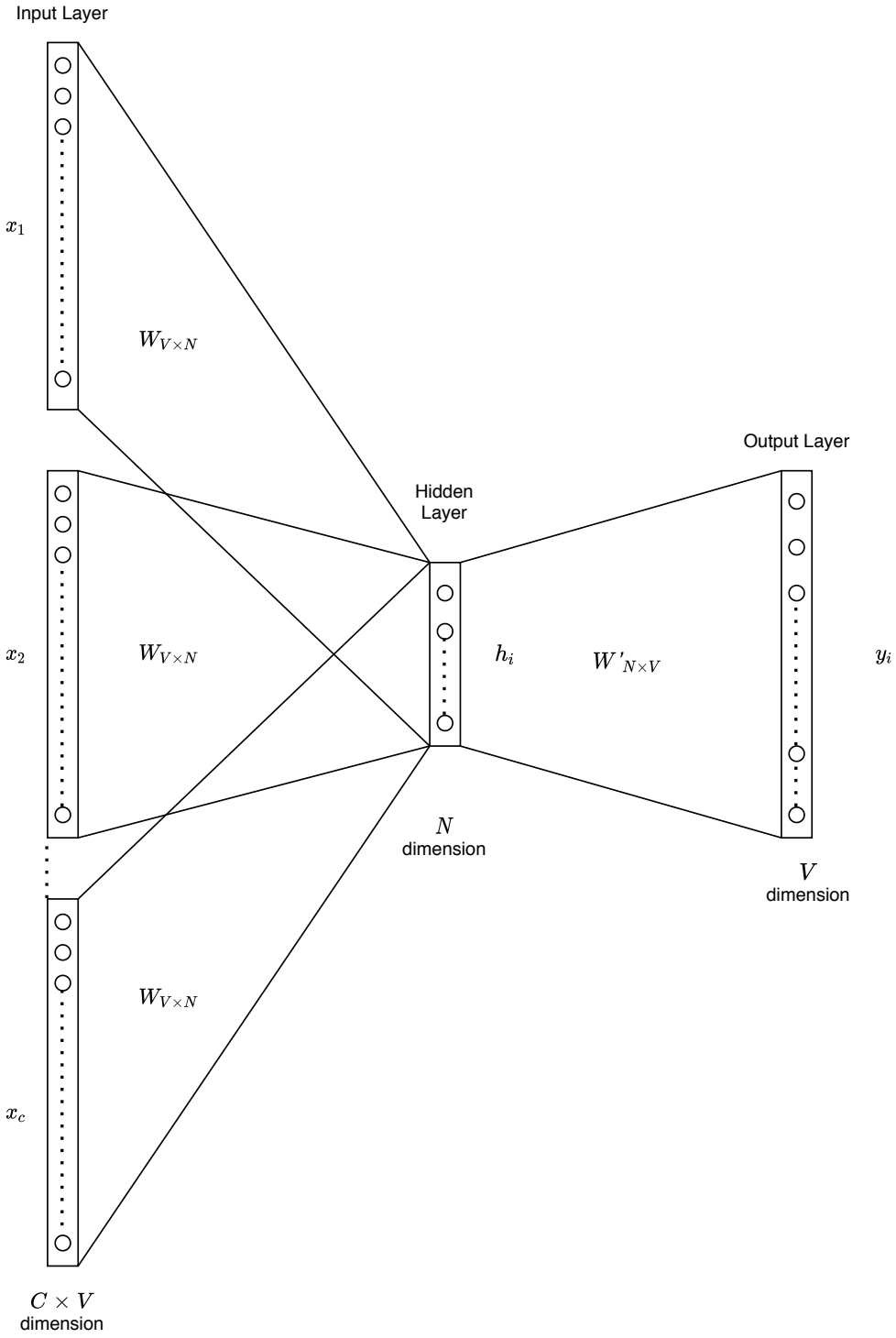

Figure 2.3: CBOW model with multiple words in the context

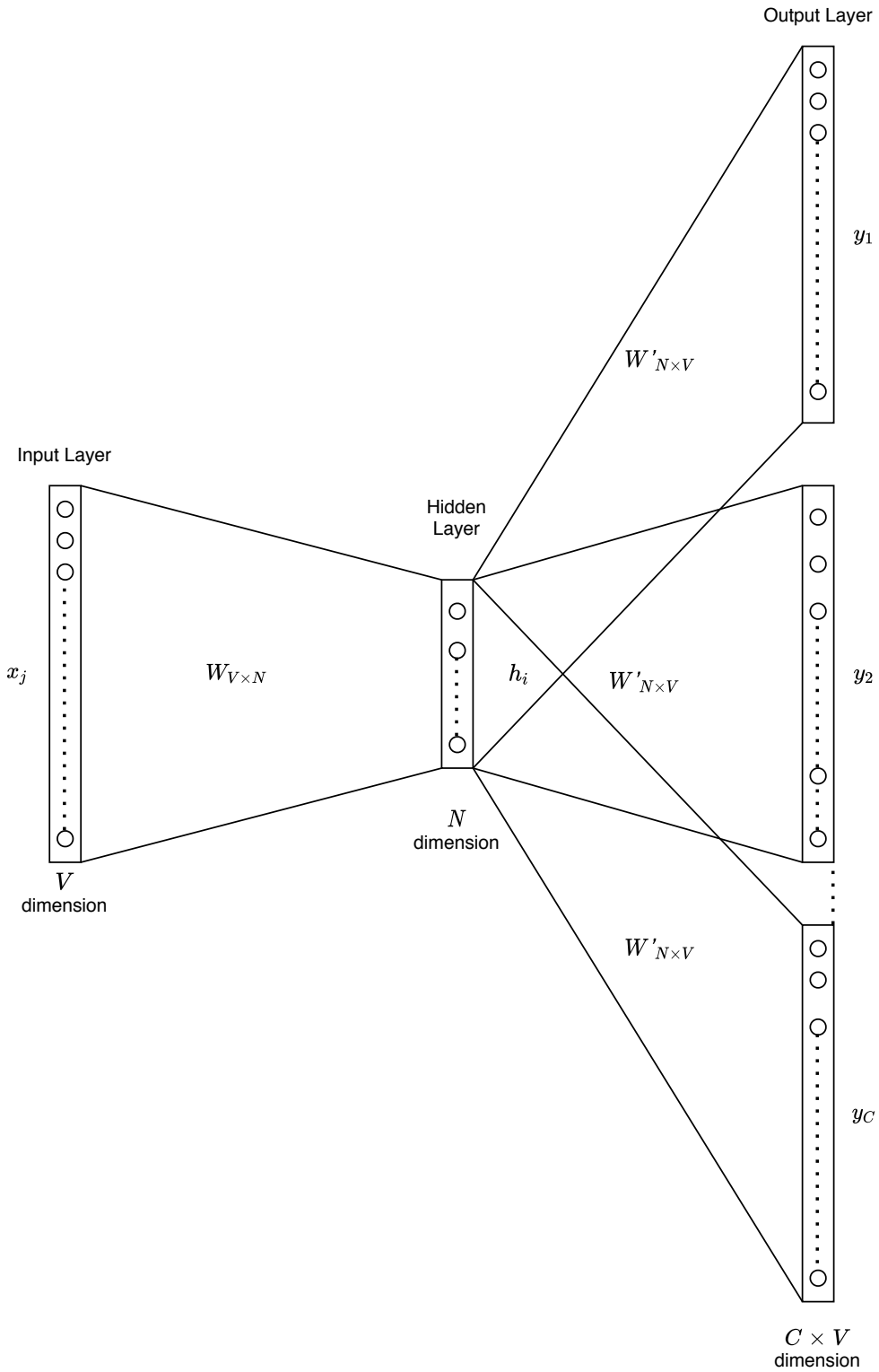

Figure 2.4: Skip-gram model for multiple context words

2.6 GloVe

GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm for obtaining vector representations for words where training is performed on aggregated global word-word co-occurrence statistics from a corpus. The resulting representation showcase interesting linear substructures of the word vector space [29].

2.6.1 Computation

Let X_{ij} = number of times word j occurs in the context of word i. Thus, the probability of such an occurrence is given by:

$$X_i = \sum_k X_{ik} \tag{2.1}$$

$$P_{ij} = P(j \mid i) = \frac{X_{ij}}{X_i} \tag{2.2}$$

The ratio of co-occurrence probabilities is then defined as:

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}}, \tag{2.3}$$

where $w_i, w_j, w_k \in \mathbb{R}^d$ are word vectors and function F gives us the correlation of the probe word w_k with word w_i and w_j. If the ratio is large, the probe word is related to w_i but not w_j, thus giving us hints on the relationships between 3 words.

F is reformulated such that it only depends on difference of two target words:

$$F((w_i - w_j), w_k) = \frac{P_{ik}}{P_{jk}} \tag{2.4}$$

Note that the arguments of F in equation 2.4 are vectors while the right-hand side is a scalar. Hence, dot product of the arguments is taken, and written as:

$$F((w_i - w_j)^\intercal w_k) = \frac{P_{ik}}{P_{jk}} \tag{2.5}$$

F should be symmetric, which require that F be a homomorphism between the groups

$(\mathbb{R}, +)$ and $(\mathbb{R}_{>0}, \times)$, i.e.

$$F((w_i - w_j)^\top w_k) = \frac{F(w_i^\top w_k)}{F(w_j^\top w_k)} \tag{2.6}$$

From equation 2.5 and 2.6, $F(w_i^\top w_k) = P_{ik} = \frac{X_{ik}}{X_i}$. If F is set to be an exponential function, $F(w_i^\top T w_k) = exp(w_i^\top w_k)$, we get:

$$w_i^\top w_k = log(X_{ik}) - log(X_i) \tag{2.7}$$

$$w_i^\top w_k + b_i + b_k = log(X_{ik}) \tag{2.8}$$

To maintain the symmetrical requirement between the words i and k, two bias terms b_i and b_k are added. Thus, word embeddings w_i and w_k are learned using equation 2.8 as the soft constraint.

In GloVe, the similarity of the hidden factors is measured between words to predict their co-occurrence count. This not only predicts the co-occurrence words, but also creates vector representations that can predict their co-occurrence counts in the corpus too.

2.7 Similarity Measure

One of the major problems addressed in this thesis is devising an efficient method of mapping sentences from one version of a privacy policy to the sentences of the successive version of the policy such that paired sentences from the two different polices have the same semantic meaning. A similarity measure takes pairs of embedding (string data converted into vectors of real numbers) as input and returns a value that represents the extent of similarity between them. We are using two similarity measures: 1) cosine similarity and 2) word mover's distance.

2.7.1 Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is defined as the cosine of the angle between them, which is same as the inner product of the same vectors normalized to both have unit length. It is a judgment of orientation and not magnitude: two vectors with the same orientation

have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. The cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. Given two vectors A and B, the cosine similarity between them is represented using a dot product and magnitude as

$$cosine\_similarity(A,B) = \frac{A \cdot B}{||A||||B||} \tag{2.9}$$

Values closer to 1 represent more similarity, while values closer to 0 represent more dissimilarity.

2.7.2 Word Mover's Distance

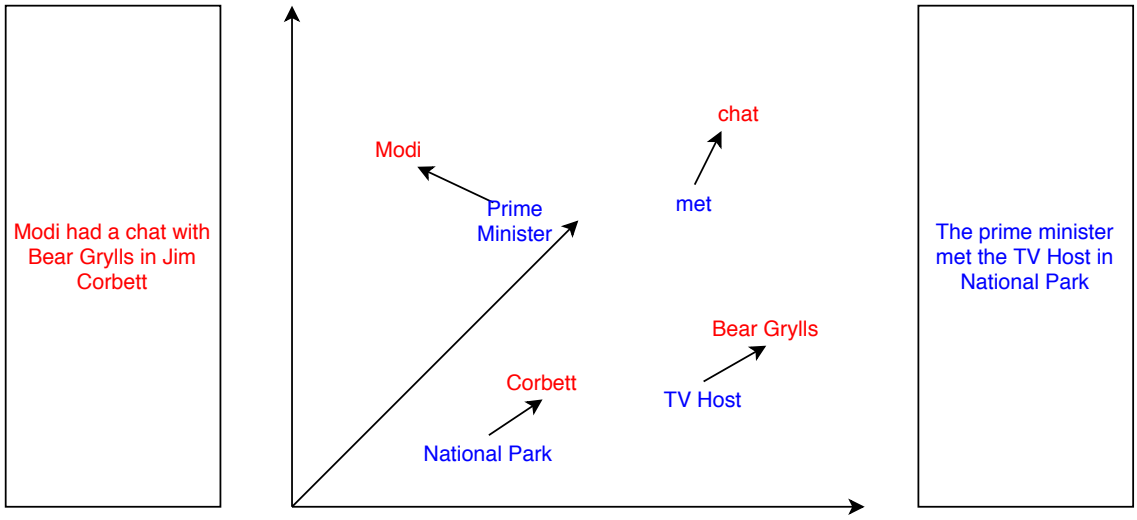

Figure 2.5: Example of WMD similarity measure in vector space

Word mover's distance (WMD) is a measure developed specifically for text similarity. It leverages the results of advanced embedding techniques like word2vec and glove, which learn semantically meaningful representations for words from local co occurrences in sentences. This preserves the semantic relationships between embedded word vectors. WMD suggests that distance between two embedded word vectors are to some degree semantically meaningful and utilizes this property to provide a measure of similarity between them. WMD treats texts as a weighted point cloud of embedded words and the similarity measure between two text A and B are given by the minimum

cumulative distance that words from text A needs to travel to match exactly the point cloud of text B [15]. Refer to figure 2.5, which shows a graphical representation of WMD measure in vector space between two texts A = *"Modi had a chat with Bear Grylls in Jim Corbett"* and B = *"The prime minister met the TV Host in National Park"*. Even though there are no common bag of words between the two sentences except the stop words, WMD incorporates the semantic relationship between the words learnt during word2vec embedding.

2.8 Machine Learning

There are many definition of Machine Learning (ML); one famous definition is based on the idea of experience:

> *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [26]*

Consider a sentiment analyzer that classifies texts into either positive or negative sentiment (T). The analyzer is trained on past data (E) to classify the text and the performance is measured by it's accuracy (P).
ML is a major subject in AI and used in conjunction with NLP to create intelligent systems that use natural language as inputs for various tasks. For example: Siri (a virtual assistant developed by Apple Inc.) takes speech as input (natural language), decipher it and perform the task assigned by the user using a system based on machine learning and natural language processing.

There are four subcategories of machine learning as illustrated in figure 2.1. These are: 1) supervised learning, 2) unsupervised learning, 3) reinforcement learning and 4) deep learning.

Supervised and unsupervised learning differ in terms of the structure of the training data used. In a supervised learning model, the algorithm learns on a labeled data set that provides an answer key that the algorithm can use to evaluate its accuracy on training data. An unsupervised model, in contrast, provides unlabeled data that the algorithm tries to make sense of by extracting features and patterns on its own.

Reinforcement learning is the training of machine learning models to make a sequence of decisions. The system learns to achieve a goal in an uncertain, potentially

complex environment. The computer employs trial and error to meet the given objective. The machine gets either rewards or penalties for the actions it performs with the goal to maximize the total reward, it is up to the model to figure out how to perform the task to maximize the reward. This type of learning is often used in games, self driving cars, robotics etc.

Deep learning (DL) is a method of solving problems by enabling a computer to infer complex patterns from simpler ones. Deep neural networks(DNN) are the basic parts of DL [9]. DNN were inspired by information processing and distributed communication nodes in biological systems. DNN are organised layer-wise, where each layer learns to transform its input data into a slightly more abstract and composite representation. Deep learning has been applied to fields such as computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering etc.

In summary, the boundaries between AI, NLP and ML are not exactly lucid, and thus should not be understood individually only. These topics are often used in conjunction with each other and thus should be studied as such. The next two sections discuss two machine learning algorithms that are analyzed, used and tested for classifying texts in this research: 1) logistic regression and 2) support vector machine.

2.9 Logistic Regression

Logistic regression is based on a mathematical model that uses probability to classify data categorically. Due to unbounded nature of linear regression, it is not suitable for classification task. A threshold is needed based on which classification can be done. Thus, making linear regression unsuitable for classification problem. This brings logistic regression into picture with prediction value strictly ranging from 0 to 1.

2.9.1 Theory

Logistic regression uses the natural logarithm function to find the relationship between the variables and uses test data to find the coefficients. The function can then predict the future results using these coefficients in the logistic equation. Logistic regression uses the theory of odds ratio to calculate the probability of a data point belonging to a class.

The odds of data point x belonging to class $y = 1$ can be defined as:

$$Odds = \frac{P(y = 1 \mid x)}{1 - P(y = 1 \mid x)} \tag{2.10}$$

$$\tag{2.11}$$

Logistic regression takes the natural logarithm of the odds to be linear with respect to x

$$\frac{P(y = 1 \mid x)}{1 - P(y = 1 \mid x)} = e^{ax+b} \tag{2.12}$$

$$\begin{aligned} P(y = 1 \mid x) &= \frac{e^{ax+b}}{1 - e^{ax+b}} \\ &= \frac{1}{1 + e^{-(ax+b)}} \end{aligned} \tag{2.13}$$

Equation 2.13 is the sigmoid function, where variable a and b are calculated using the training set.

2.9.2 Multinomial Logisitic Regression

Multinomial logistic regression is used when the number of categories are more than two. The fundamentals are based on the same principles as of logisitic regression. If the multinomial logit is used to model choices, it relies on the assumption of independence of irrelevant alternatives (IIA), which is not always desirable. This assumption states that the odds of preferring one class over another do not depend on the presence or absence of other "irrelevant" alternatives. This means that the choice of K alternatives or classes is modeled as a set of $K - 1$ independent binary choices in which one class is chosen as "pivot" and other $K - 1$ classes are compared against it. As our sentence classification task had more than two categories, an implementation provided by *scikit-learn* package was used for this research.

2.10 Support Vector Machine

Support vector machine (SVM) is a simple machine learning algorithm with the potential to classify data points (in our case text) with significant accuracy and less

computational power. It can be used for both classification and regression tasks, but most widely used for classification objectives. Support vector machines were originally designed for binary classification. Several methods have been proposed that realizes a multi-class by combining several binary classifiers.

2.10.1 Objective and Theory of Binary SVM Classifier

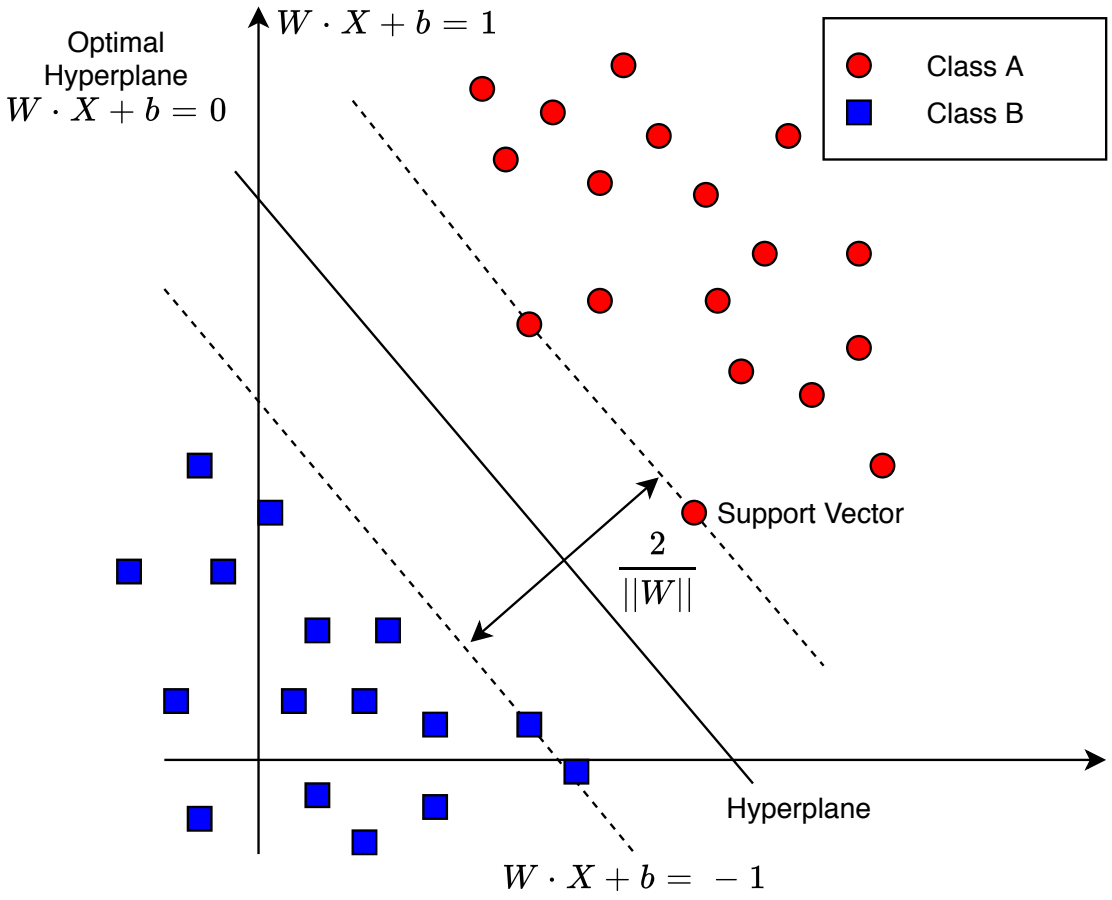

Figure 2.6: Classification of data by support vector machine (SVM)

The objective of a support vector machine algorithm is to find a hyperplane that separates the data points (in an N-dimensional space) into two classes, where N is the number of features. In case of text represented as vectors of real numbers, N will refer

to the dimension of the vector. To separate the two classes of data points, the objective is to find a plane that separates data points of both the classes with maximum margin. Maximizing this margin instills more confidence on the future predictions.

Hyperplanes are the decision boundary supported by the support vectors, which are data points that are closer to the hyperplane. Support vectors not only influence the position and orientation of the hyperplane but also helps in maximizing the margin. In short, support vectors are the points that build a SVM classifier. If the number of features is 2 i.e $N = 2$, then the hyperplane is simply a line. If $N = 3$, then the hyperplane becomes a two-dimensional plane.

For simplicity, let us consider all the data points to be two dimensional. Then the hyperplane is a line and can be written mathematically as:

$$y = ax + b \tag{2.14}$$
$$ax + b - y = 0$$

Let $X = (x, y)$ and $W = (a, -1)$, then the vector form of the hyperplane is

$$W \cdot X + b = 0. \tag{2.15}$$

For a given point X_i from the training set, if it belongs to one class (say class A) then it should follow $W \cdot X_i + b \geq 1$. If it belongs to the other class (say class B) then $W \cdot X_i + b \leq -1$ should be followed.

Support vectors are represented as $W \cdot X + b = 1$ and $W \cdot X + b = -1$. Thus, the margin $\frac{2}{||W||}$, refer figure 2.6. SVM finds the support vectors and hyperplane, or find W and b, such that $\frac{2}{||W||}$ is maximized and $Y_i(W \cdot X_i + b) \geq 1$ where $Y_i = 1$ or -1 representing each of the two classes (A and B) and $i \in \{1, 2, \ldots L\}$ with L as the size of the training set. Even though the discussion above is for two dimensional data, the same theory is applicable for any N-dimensional data set.

2.10.2 Non-Linearly Separable Data

It is not always the case that raw data is linearly separable. However, adding one extra dimension can make it separable. Figure 2.7 shows how adding a dimension representing the square of magnitude of the data points can make them linearly separa-

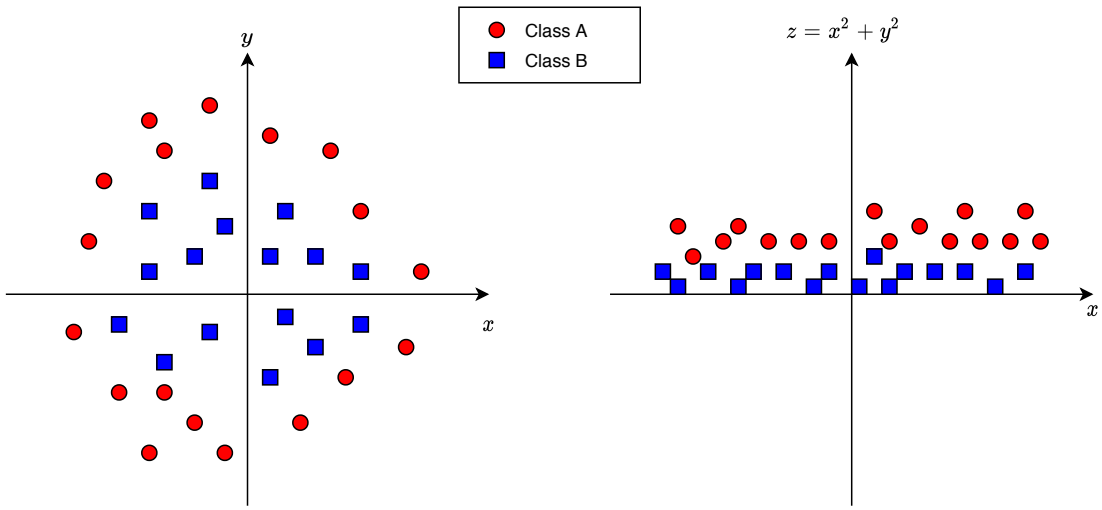

Figure 2.7: Example of transforming non-linearly separable data using kernel trick

ble. This is known as Kernel trick. Most implementations of SVM provide the option of choosing a kernel type. For example, *svm.SVC* from *sklearn* have the option of using 'linear', 'poly', 'rbf', 'sigmoid' or a 'precomputed' kernel.

2.10.3 Multi class SVM

As mentioned before, support vector machines (SVM) were originally designed for binary classification. The theory discussed above also consider a two class classification. But it can be extended to multiple classes using either of these methods : one-against-all, one-against-one and DAGSVM.

In this research we have used a multi class SVM using an implementation the one-against-one from *scikit-learn* [1]. $\frac{nClasses \times (nClasses-1)}{2}$ classifiers are constructed, where $nClasses$ is the number of classes and each one trains data from two classes. The multi-class classification is transformed into one binary classification problem per each pair of classes.

2.11 Technologies

The following section will describe the technologies that are used in this thesis implementation. Python packages and libraries used in this research are described below.

[1] https://scikit-learn.org/stable/modules/svm.html

2.11.1 NumPy

NumPy brings the speed of compiled code to Python, with it's core built on well optimized C code. *NumPy* provides the functionality of creation of N-dimensional arrays and provides fast versatile array computation methods. It also offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more [2].

2.11.2 Pandas

Pandas is an open source library that provides custom data structures such as dataFrames and series [3]. These structures are indispensable while working with large data, especially in analysis.

2.11.3 MatplotLib

Matplotlib is a library for making 2D plots of arrays in Python. Although it has its origins in emulating the MATLAB graphics commands, it is independent of MATLAB, and can be used with Python in an object oriented way. *Matplotlib* is written primarily in pure Python and makes heavy use of *numpy* and other extension code to provide good performance even for large arrays [4]. *Matplotlib* is used in this thesis for data analysis and visualization while comparing conceiving tools and techniques for matching sentences.

2.11.4 NLTK

Natural language toolkit (NLTK) is a leading platform for building Python programs to work with human language data [5]. *NLTK* libraries are used in this thesis for tokenization, stemming, lemmatization, tagging and parsing. Also, *NLTK* provided corpora such as *WordNet* and resources like stop word list which are used for processing text data.

[2] https://numpy.org/
[3] https://pandas.pydata.org/about.html
[4] https://matplotlib.org/3.2.1/users/history.html
[5] https://www.nltk.org/

2.11.5 Scikit-learn

Scikit-learn, or often referred to as *sklearn*, is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and uns-upervised problems [28]. *Sklearn* provides flexible and efficient tools for classification, regression, clustering, dimensionality reduction, preprocessing and model selection[6]. For this thesis, label encoder from *sklearn* is used in preprocessing. It also provides *tfidf vectorizer* for feature extraction. Most importantly, *sklearn* provided tools for classifiers that are tested, analyzed and used in this research, such as SVM and multinomial logistic regression. Lastly, all the evaluation metrics used are also provided by *sklearn*.

2.11.6 Gensim

Gensim is a language processing toolkit for Python. It is designed to work with large corpora and provides various scripts for topic modeling. *Gensim* provides implementation of word2vec and glove.

2.11.7 spaCy

spaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython [7]. It features NER, POS tagging, dependency parsing, word vectors and more, which are used in this research for detecting changes between matched sentences.

[6] https://scikit-learn.org/stable/
[7] https://spacy.io/

Tool/Package	Purpose
Python	Primary programming language
NumPy	Data manipulation and arithmetic operations
Pandas	Data analysis and representation
MatplotLib	Data visualization and analysis
NLTK	Text processing
Sklearn	Machine learning, data encoding and performance evaluation
Gensim	Word2vec and glove encoding
spacy	Pos tagging, tree parsing and operation

Table 2.1: List of tools/packages and their application in the research

3. Data Description

This chapter is dedicated to the description of the data that has been used in our research. We have used 1) OPP-115 corpus and 2) privacy policies.

3.1 OPP-115 Corpus and Annotation Scheme

The (Online Privacy Policies) OPP-115 corpus is a collection of 115 website privacy policies with annotations that specify data practices in the text. Each privacy policy was read and annotated by three graduate students in law [37]. The task of classifying sentences before matching them requires the sentences to be compliant to categories that properly represent their semantic. OPP-115 corpus contains detailed annotations for the data practices described in the collection of website privacy policies. At a high-level, these annotations fall into one of the ten data practice categories, developed by a team of legal experts:

1. First Party Collection/Use: how and why a service provider collects user information.
2. Third Party Sharing/Collection: how user information may be shared with or collected by third parties.
3. User Choice/Control: choices and control options available to users.
4. User Access, Edit, & Deletion: if and how users can access, edit, or delete their information.
5. Data Retention: how long user information is stored.
6. Data Security: how user information is protected.

7. Policy Change: if and how users will be informed about changes to the privacy policy.

8. Do Not Track: if and how do not track signals for online tracking and advertising are honored.

9. International & Specific Audiences: practices that pertain only to a specific group of users (e.g., children, residents of the European Union, or Californians).

10. Other: additional privacy-related information not covered by the above categories.

In OPP-115 corpus annotations, each policy has an associated CSV file containing the annotation data for that policy. Each file contains data from all the three workers. Each row of the CSV file contains the information for a single data practice and have the following attributes:

1. Annotation ID: a globally unique identifier for the data practice.

2. Batch ID: name of the batch in the annotation tool.

3. Annotator ID: identifies the annotators.

4. Policy ID: identifier for the policy.

5. Segment ID: sequential identifier of the policy segment.

6. Category name: assigned category for the segment.

7. Attribute-value pairs: JSON object containing detailed annotations.

8. Policy URL: source URL for the policy.

9. Date: revision date of the policy.

The attribute-value dictionary, at its highest level, consists of keys (strings that correspond to attribute names) that map to nested dictionaries. The nested dictionaries have keys that specify the selected text, its location in the segment, and the value associated with the attribute. For building the classifier, sentence texts were extracted from the attribute-value pairs under the key *selectedText*. Corresponding category of the sentence was retrieved from the category-name attribute.

3.2 Privacy Policies

The most important data used in this research are privacy policies. OPP-115 corpus contains only one policy per website. We require multiple versions of a policy. Using the Wayback Machine archive [1], we collected the available versions of the policy for six six global technology organizations. For example, Facebook policies from the year 2005 to 2018 were collected. These policies are collected as HTML pages, then processed to extract the texts from the pages without any HTML tags. Facebook policy texts are the primary data used for our research. The revisions month and dates of the 18 Facebook policies are June 2005, February 2006, May 2006, October 2006, May 2007, September 2007, December 2007, November 2008, December 2009, October 2010, December 2010, January 2011, September 2011, December 2012, November 2013, January 2015, September 2016 and October 2018. We also collected all versions of policies for the following organizations: WhatsApp (6 policies from 2009 to 2019), Twitter (14 policies from 2007 to 2020), Google (27 policies from 1999 to 2017), LinkedIn (5 policies from 2013 to 2020) and Snapchat (9 policies from 2017 to 2020).

3.3 Other Data

For training the glove and word2vec vectorizers, corpora glove.840B.300d is used [2]. This corpora contains 840 billion tokens, 2.2 million vocabulary and 300 dimensional cased vectors created using common crawl of the web. This corpora is available under the Public Domain Dedication and License [3].

[1] https://archive.org/web/

[2] http://nlp.stanford.edu/data/wordvecs/glove.840B.300d.zip

[3] https://opendatacommons.org/licenses/pddl/1-0/

4. Sentence Classifier

The first task of the research is to build a sentence classifier using the OPP-115 corpus that can be used to automatically annotate the texts of the policies according to expert defined categories at a sentence level. The final objective of this research is to present user-relevant changes between two versions of a privacy policy which requires a sentence classifier. Otherwise during sentence matching, due to sheer volume of text data, not only comparing one sentence of the previous policy to every other sentence of the next policy becomes computationally expensive but also the chances of bad match increase. Limiting the match search in the same category is faster and produces better result. For this task, existing annotations of the OPP-115 corpus is leveraged to build a classifier that can automate the process of categorizing the texts. This part of the research only uses the OPP-115 corpus for training and testing.

4.1 Text to Tokens

As mentioned in chapter 3.1, only sentences (selected texts) and their corresponding classifier attributes are loaded into a *pandas dataFrame* for building the sentence classifier. Each sentence is preprocessed and converted into tokens first using Algorithm 1. Algorithm 1 takes the entire column of sentences in the *dataFrame* as input and returns a column of list of tokens corresponding to each sentence. The first step is to drop the row from the *dataFrame* with empty or null sentences. Then for each sentence all uppercase characters in the string are converted into lowercase. Using *word_tokenize* from *nltk.tokenize*, sentences are broken down into a list of words or tokens. These tokens are further preprocessed by removing english stop words and non alphabetic words. Finally, using *WordNetLemmatizer* from *nltk.stem* and part of speech tagging from *nltk*, these tokens are lemmantized and added to the final list of tokens for that sentence. The process is repeated for all the sentences. Refer to 2.11.4

and 2.2 for more details on *nltk* and preproccesing tools used.

Algorithm 1: Sentences to tokens

```
input  : sentence List
output: list of tokens
tokenList ← [] ;
foreach sentence s of the sentenceList do
    if s.length == 0 then
        drop corresponding row from dataFrame;
        continue;
    end
    s = s.lower();
    tokens ← word_tokenize(s);
    final_tokens ← [];
    foreach word, tag in tokens do
        if word not in stopwords.words('english') and word.isalpha()
         then
            final_tokens.add(WordNetLemmatizer(word, tag));
        end
    end
    tokenList.add(final_tokens);
end
```

4.2 Feature Extraction

The tokens that are created are needed as floating point values for the machine learning algorithms. Due to the large number of different words in these policies, tf-idf vectorization is selected for this encoding or feature extraction. After looking at the frequency distribution of all tokens, *max_feature* for the tf-idf vectorization is selected as 5000, which creates a feature matrix of 5000 most frequent words. If this parameter is selected as *none,* then the feature matrix will consider the exceptionally rare terms as well. This would have resulted in high weight assignments to these rare words and resulted in biased prediction of categories in presence of rare words. *TfidfVectorizer* from *sklearn.feature_extraction.text* was used for this. It was trained on the documents in OPP-115 corpus. Instead of using a different corpora, the OPP-115 corpus was used

so that privacy policy related tokens are assigned appropriate weights. Using a different corpora for this would have resulted in *none* weights for policy related words due to the *max_feature* parameter and rarity of these terms in general documents. The trained vectorizer was then used to encode our text data (tokens) into floating point vectors. Tf-idf is discusssed in details in section 2.4. The trained vectorizer was also saved for future transformation of the policies used in this research for consistency in text encoding.

4.3 Prediction and Evaluation

The encoded vectors and their corresponding categories from the OPP-115 corpus were randomly split into a 70% training set and a 30% testing set. Categories are converted into integer values using *sklearn.preprocessing.LabelEncoder*, i.e our 10 categories defined in 3.1 were assigned values from 0 to 9. The training set is used to train two supervised learning models: logistic regression and support vector machine, evaluated using the testing set. This process was repeated 10 times for both models, each time randomly splitting the training and testing set according to the mentioned ratio. Predictive models used, were selected based on already established methods [18] for automated annotations of privacy policies.

For logistic regression *sklearn.linear_model.LogisticRegression* is used with *liblinear* solver, which uses one-vs-rest scheme for multinomial classification. Rest of the parameters were set to default. Logistic regression performed with a precision of 0.59, recall of 0.77 and F1-score of 0.65. Here, precision is calculated as the sum of true positives across all classes divided by the sum of true positives and false positives across all classes. Recall is calculated as the sum of true positives across all classes divided by the sum of true positives and false negatives across all classes. F1-score is (2 × Precision × Recall) divided by (Precision + Recall).

SVM classifier is realized using *sklearn.svm.SVC,* an *libsvm* implementation. The model was trained with the following parameters: *C* was set to 1 , thus opting for a larger margin separating hyperplane at the cost of more potential misclassification; *kernel* was set to linear and other parameters were set to default. The SVM classifier had the precision of 0.74, recall of 0.83 and F1-score of 0.78.

Evaluation of both logistic regression and support vector machine were done with the OPP-115 corpus. Our observed results of the evaluation were on par with results

Model	Precision	Recall	F1-score
Logistic Regression	0.59	0.77	0.67
Support Vector Machine	0.74	0.83	0.78

Table 4.1: Evaluation results (precision/recall/F1-score) for logistic regression and support vector machine

from already established tools using these machine learning models [18] and validates the proficiency of a SVM classifier and it's superiority over logistic regression in automated annotation of privacy policies. Thus, SVM classifier was finalized as the classification tool to be used for the rest of the work done in this research. The next section discusses results and observations from using the SVM classifier (trained on OPP-115 corpus) on Facebook privacy policies, which serves as the primary data set for majority of this research.

4.4 Results and Discussion

The trained SVM model is evaluated using the OPP-115 corpus, but we wanted to see how it performs for our collected Facebook policies. A ground truth for the Facebook privacy policies from September 2016 and October 2018 is created by manually categorizing each sentence of the two policies. Then, using the trained classifier, these two policies were annotated and the ground truth was used to evaluate the classifier's performance. The precision, recall and F1-score for classification of both the policies are shown in table 4.2. Facebook privacy policy from September 2016 was annotated with a performance almost consistent to observed values during evaluation of the SVM model with OPP-115 corpus. But the policy from October 2018 performed poorly in terms of precision, recall and F1-score, which begged the question why such a drastic drop in performance for this policy. Creating ground truth for all the different versions of Facebook policies would have been a very labour intensive task. It was thus decided that instead of manually annotating each policy and evaluating the classifier on those, a deeper analysis using the available ground truth of the two policies can also give us some interesting insight and maybe give us the reason behind the poor annotation of October 2018 privacy policy. With that idea in mind, confusion matrices for the two Facebook policies were plotted for further analysis.

Policy	Precision	Recall	F1-score
Facebook September 2016	0.65	0.85	0.74
Facebook October 2018	0.48	0.58	0.50

Table 4.2: Classification results (precision/recall/F1-score) for sentences from Facebook privacy policies

4.4.1 Confusion Matrix and Analysis

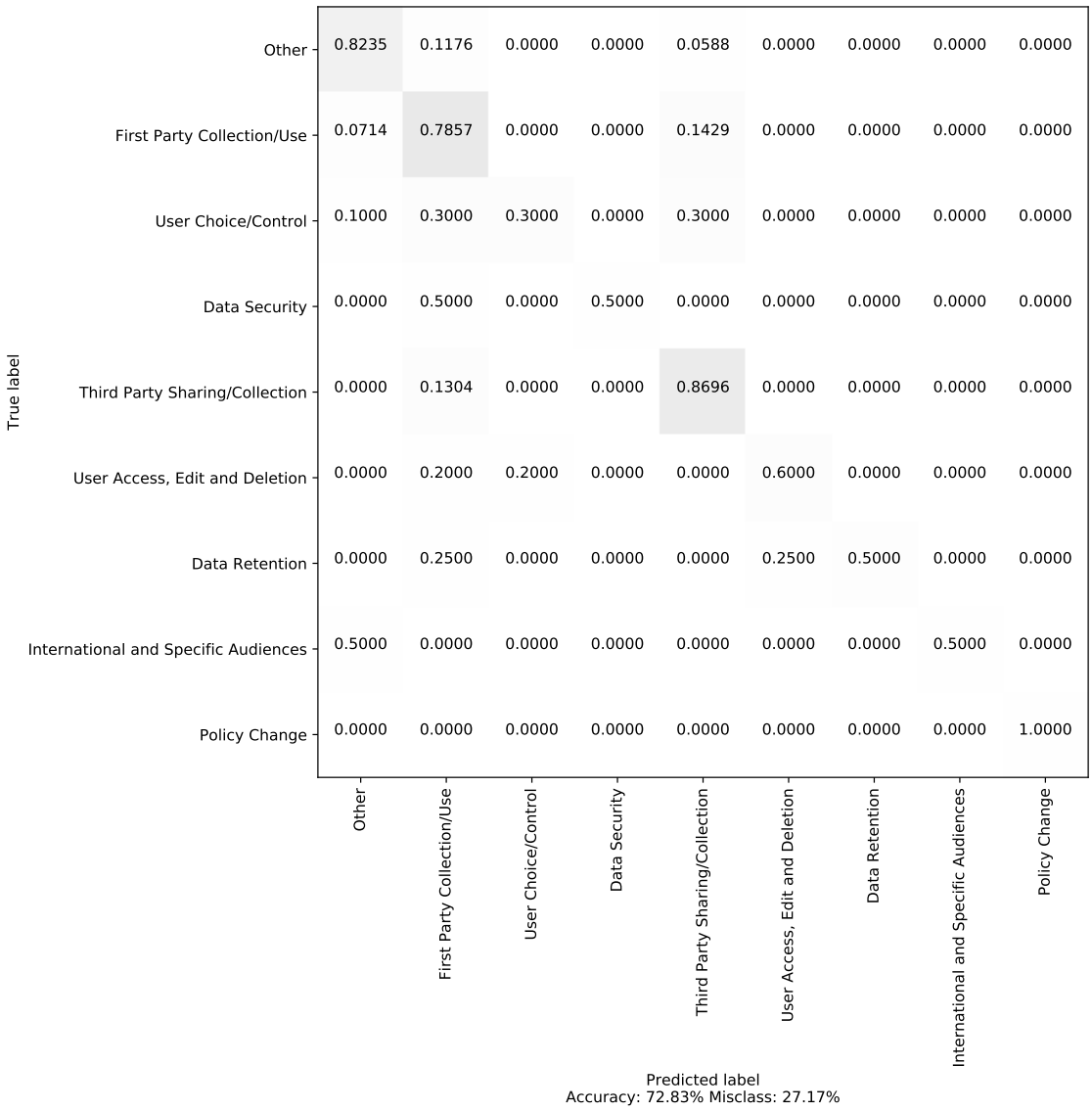

Figure 4.1: Confusion Matrix for Facebook (September 2016) annotation

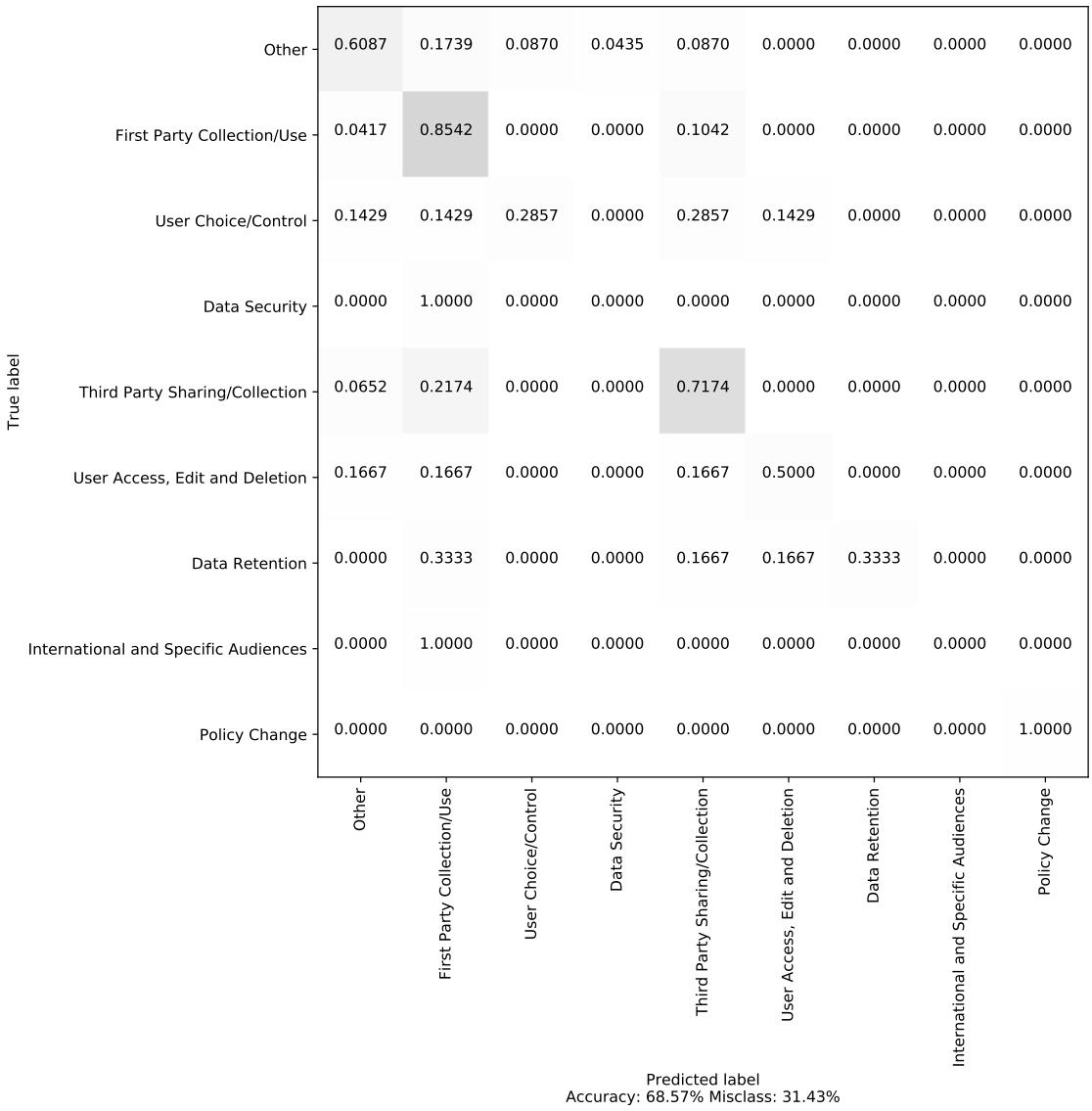

Figure 4.2: Confusion Matrix for Facebook (October 2018) annotation

A more detailed analysis of annotations generated by our SVM classifier was conducted by first plotting confusion matrices of Facebook privacy policies from September 2016 and October 2018 shown in figure 4.1 and figure 4.2 respectively. The confusion matrices showed major misclassification of sentences that should have belonged to categories “User Choice/Control” and “Data Security” in both policies. Taking examples from these categories some observations were made.

“We also offer easy-to-use security tools that add an extra layer of security to your account”

should have been classified into “Data Security” but was categorized into “First Party Collection/Use”. After going through similar examples of misclassification, it was observed that presence of certain tokens which have higher association with other categories creates a bias in classification. In the example mentioned above, presence of words “we”, “use” and “account” which are commonly observed in sentences belonging to“First Party Collection/Use” led to it’s misclassification. Other examples also showed similar trends where presence of certain tokens have a higher weight towards a certain category leading to misclassification. Some sentences such as

> *“We collect information from or about the computers, phones, or other devices where you install or access our Services, depending on the permissions you have granted”*

that can technically belong to more than one category (either “User Choice/Control” or “First Party Collection/Use” in this case but predicted as “Other”), often get wrongly classified. This implies that the ambiguity in interpretation of sentences while annotating texts during the creation of OPP-115 corpus, has been reflected in the trained classifier, leading to a fuzzy-logic-like classification in some cases. The same obser-

Category	Frequency
First Party Collection/Use	8,956
Third Party Sharing/Collection	5,230
Other	3,551
User Choice/Control	1,791
Data Security	1,009
International and Specific Audiences	941
User Access, Edit and Deletion	747
Policy Change	550
Data Retention	370
Do Not Track	90

Table 4.3: Frequency of sentences category wise in the OPP-115 Corpus (taken from [37])

vations were made for sentences belonging to categories other than “User Choice/-Control” and “Data Security”, which were misclassified. But the rate of such misclassification was considerably less for certain categories like “Other” , “First Party Collection/Use” and “Third Party Sharing/Collection”. As reported in [37] “Other” , “First

Party Collection/Use" and "Third Party Sharing/Collection" had the highest frequency of sentences in the corpus, shown in table 4.3. The higher frequency of sentences or data points belonging to these three categories clearly have played a major role in a better classification performance for them. Also, no sentence was found in both September 2016 and October 2018 Facebook policies belonging to the "Do Not Track" category, reflecting the statistics mentioned in [37] and in table 4.3. These observations led to further categorical analysis of the collected policies to see if a similar trend of distribution of sentences categories is observed or not.

4.4.2 Categorical structure

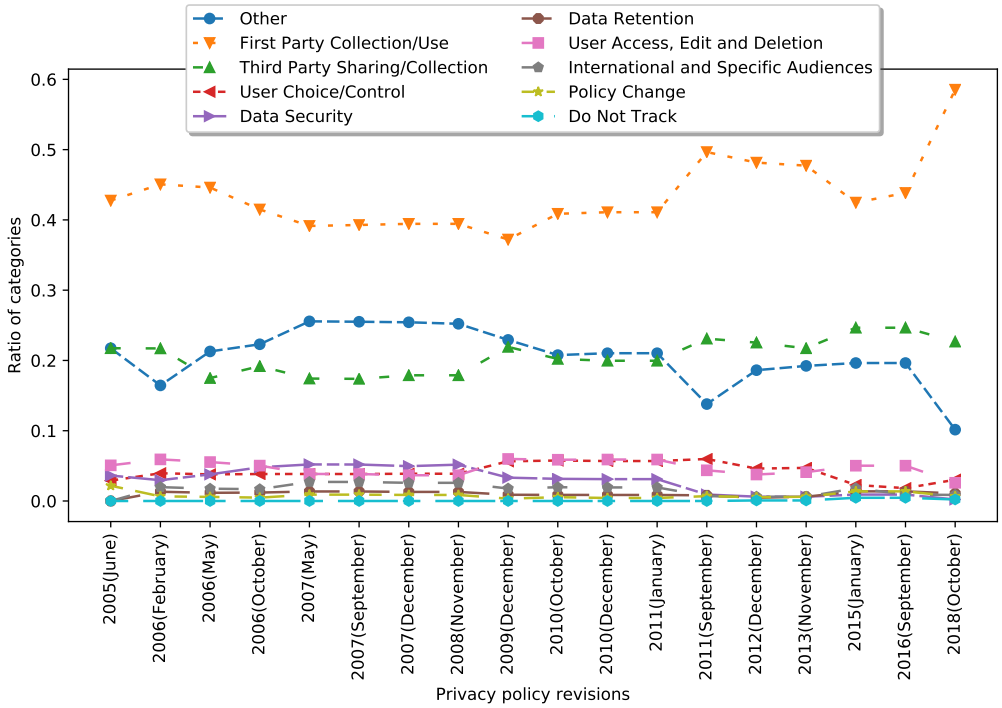

Figure 4.3: Categorical structure of Facebook privacy policies

One of the observations made during the deeper analysis of Facebook privacy policies using confusion matrices was the non-uniform distribution of sentences among all the categories. Categories "Other", "First Party Collection/Use" and "Third Party Sharing/Collection" had noticeably higher sentences than the other categories in the OPP-115 corpus, which not only made the implemented classifier more efficient for these categories but also gave us some interesting insight into the composition of most

policies. We wanted to leverage the created sentence classifier and validate whether the same trend is followed through different revisions of the same policy. To analyze that, first for each individual version of the policy we categorized the sentences using our classifier. Then, the ratio of the number of sentences found in a particular category to the total number of sentences in the policy was computed. This is done for each of the 10 categories. The same process was repeated for all the versions of a policy. The computed ratios are then plotted on a graph to observe and analyze the overall trend of categorical structure in a policy.

We began this analysis by first taking the policies from Facebook. The ratio of each category throughout each version of the policy is plotted and shown in figure 4.3. It reflects the same statistics presented in [37], with categories "Other", "First Party Collection/Use" and "Third Party Sharing/Collection" forming the major portion of each policy. Most sentences belong to the "First Party Collection/Use", which was also the case with the OPP-115 corpus. This shows that privacy policies are mostly focused on how user data is collected and used by the company. The other two major categories "Third Party Sharing/Collection" and "Other" interchangeably have the second and third highest ratio in the documents. How a company shares user data with third party services or companies have been a major concern for many users, and these high ratio of sentences for "Third Party Sharing/Collection" show that privacy policies have tried to address these user-concerns. "Other" category represents sentences which can not be confidently categorized into other categories due to lack of clear interpretation. The high frequency of "Other" category sentences both in the Facebook policies and the OPP-115 corpus shows that how ambiguous and incomprehensible the nature of these documents can be, which is not only a concern but also a major motivation behind this research.

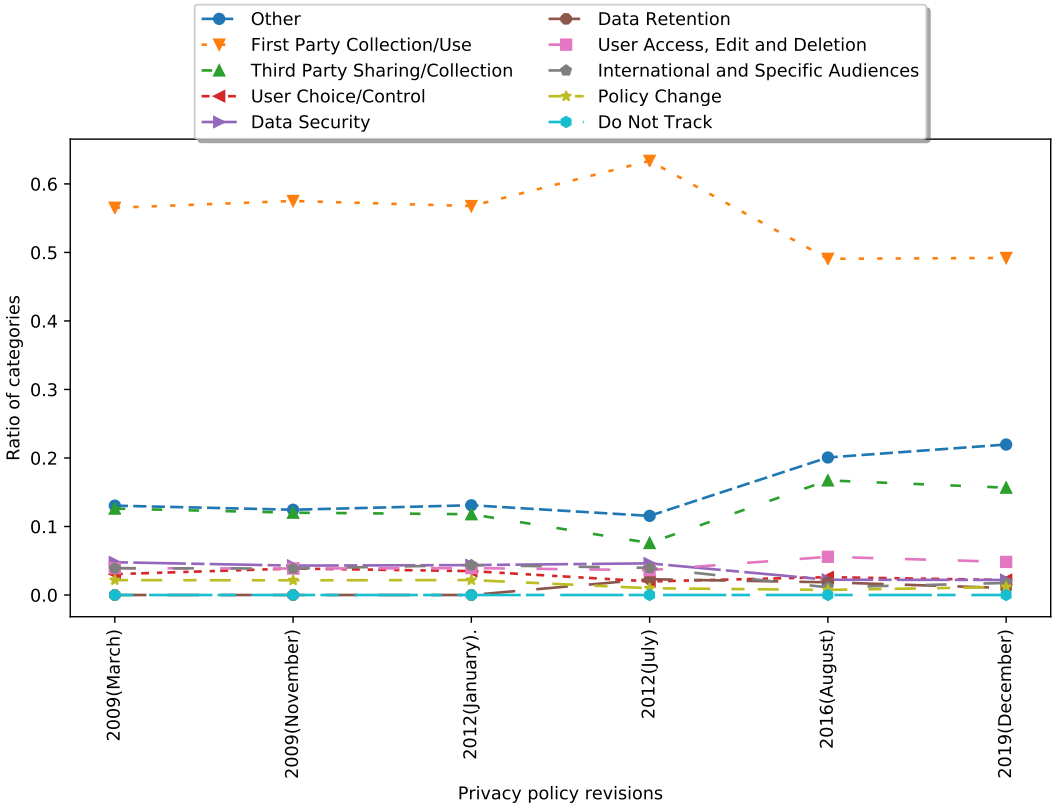

Figure 4.4: Categorical structure of WhatsApp privacy policies

We also plotted the same graph for WhatsApp and Twitter privacy policies shown in figure 4.4 and figure 4.5 respectively, which also followed the same trend of structure. Categories “User Choice/Control”, “Data Security”, “Data Retention”, “User Access, Edit and Deletion”, “International and Specific Audiences”, “Policy Change” and “Do Not Track” had less than 10 percent contribution each in all the privacy policies of Facebook, WhatsApp and Twitter. Sentences for the category “Do Not Track” is almost non-existent. This analysis gave us a deeper insight into the overall structure of these policies, helping us understand their composition at a categorical level.

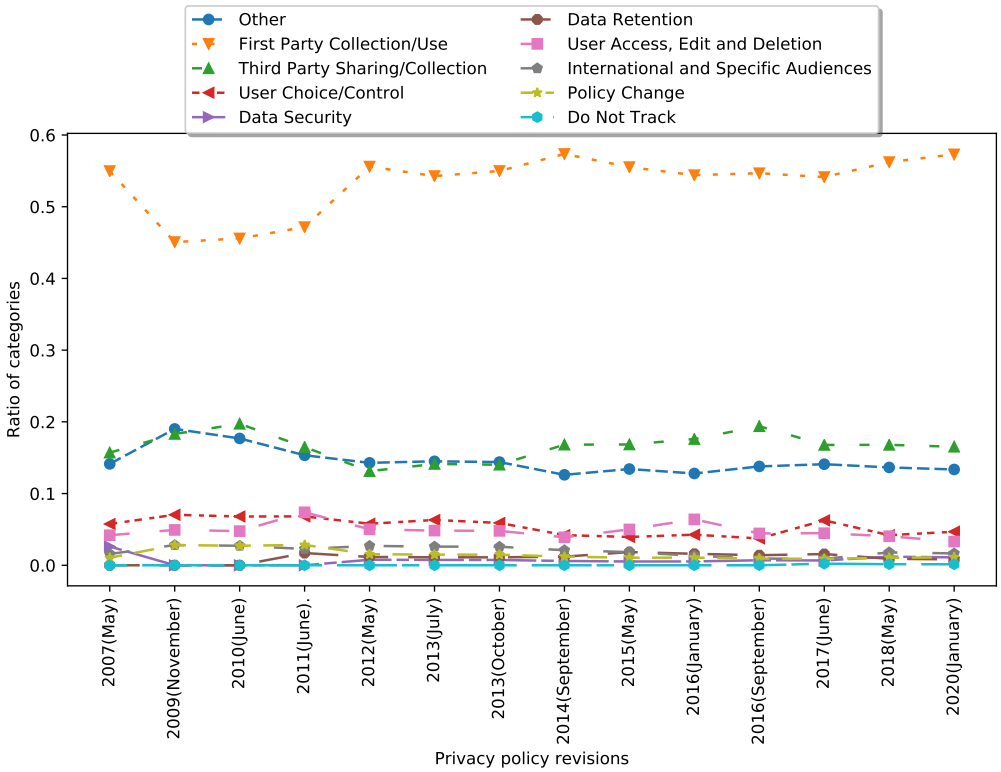

Figure 4.5: Categorical structure of Twitter privacy policies

5. Sentence Matching

The primary goal of this research is to develop tools for mapping sentences of a privacy policy to the sentences of the successive version or revision of that policy. This mapping should generate pairs of sentences such that both sentences have the same semantics and help detect user-relevant changes between them. We need a good match, otherwise it won't make sense to detect changes between a pair of sentences if they don't mean the same thing. It is very important that the semantic relationship between the words is preserved for a good match. Word2vec and gloVe embedding are potential techniques that we evaluate and analyze to finalize one for this task. On top of that, the similarity measure used also plays a pivotal role. Hence, both cosine similarity and word mover's distance are also evaluated in permutation with word2vec and gloVe. This chapter is dedicated to the study of the above mentioned techniques and finalizing a method to generate pairs of matched sentences from two different revisions of a privacy policy. SIF (Smooth Inverse Frequency) embedding on top of word2vec and glove embedding is also evaluated and discussed in this chapter.

5.1 Similarity Measures

Before evaluation of embedding methods, generic similarity measures are first defined for easier evaluation of embedding models. These methods are: 1) cosine similarity and 2) word mover's distance. Both the methods take two sentences as inputs and returns a similarity measure between them. Other arguments of the methods are *model* which can be either *glove* or *word2vec*, and *use_stoplist*, which is a boolean that determines whether stop words should be included in the vector word embedding or not. These methods are designed for easier evaluation of the similarity measures in combination with the embedding methods.

5.1.1 Cosine Similarity

Method cosine_sim shown in listing 5.1 is built on top of *cosine_similarity* from *sklearn.metrics.pairwise*. Cosine similarity is discussed in section 2.7.1. Arguments *sent1* and *sent2* are the two sentence strings for which similarity measure is being computed. Argument model can be either glove or word2vec and *use_stoplist* eliminates the stop words if set to True. Argument *doc_freqs* is for a dictionary of tokens and their respective frequencies, if provided these token frequencies are used to calculate the weights for *np.average* (weight for each vector during average computation). Weights are calculated using:

$$weight(token) = frequency(token) \times log\left(\frac{N}{doc\_freqs.frequency(token)} + 1\right) \tag{5.1}$$

$$frequency(token) = \text{frequency of the token in the input sentence}$$

$$N = \text{sum of frequency of all the tokens in the } doc\_freqs$$

$$doc\_freqs.frequency(token) = \text{frequency of the "token" in the } doc\_freqs$$

Equation 5.1 is implemented in lines 22-25 of the Python code in listing 5.1. For this research, *doc_freqs* dictionary is created using frequency files found in NLP-town github repository [1]. NLP-town is an online platform that provides resources, consultancy and information on all aspects of natural language processing. After creating the embedding for the sentences, method *cosine_sim* returns the cosine similarity between the two embeddings.

Listing 5.1: Python code for computing cosine similarity between two sentences

```
1  def cosine_sim(sent1, sent2, model=None, use_stoplist=False, doc_freqs=None):
2      stop_words = set(nltk.corpus.stopwords.words("english"))
3      # N is assigned the sum of frequency of all the tokens in the doc_freqs file
4      if doc_freqs is not None:
5          N = doc_freqs["NUM_DOCS"]
6      tokens1 = [token.lower() for token in nltk.word_tokenize(sent1)]
7      tokens2 = [token.lower() for token in nltk.word_tokenize(sent2)]
8
9      if use_stoplist:
10         tokens1 = [token for token in tokens1 if token not in stop_words]
```

[1] https://github.com/nlptown/nlp-notebooks/find/master

```
        tokens2 = [token for token in tokens2 if token not in stop_words]

    tokens1 = [token for token in tokens1 if token in model]
    tokens2 = [token for token in tokens2 if token in model]

    if len(tokens1) == 0 or len(tokens2) == 0:
        return 0

    freqs1 = Counter(tokens1)
    freqs2 = Counter(tokens2)

    wts1 = [freqs1[token] * math.log(N/(doc_freqs.get(token, 0)+1))
                for token in freqs1] if doc_freqs else None
    wts2 = [freqs2[token] * math.log(N/(doc_freqs.get(token, 0)+1))
                for token in freqs2] if doc_freqs else None

    embedding1 = np.average([model[token]
                        for token in tokfreqs1], axis=0, weights=wts1)
    embedding1 = embedding1.reshape(1, -1)
    embedding2 = np.average([model[token]
                        for token in tokfreqs2], axis=0, weights=wts2)
    embedding2 = embedding2.reshape(1, -1)

    sim = cosine_similarity(embedding1, embedding2)[0][0]
    return sim
```

5.1.2 Word Mover's Distance

Method wmd shown in listing 5.2 is similar to *cosine_sim,* but it doesn't have the *doc_freqs* parameter. Since word mover's distance leverages the semantic relationship in the embeddings from word2vec and glove, normalization of word embeddings using frequency statistics is rhetorical. Both glove and word2vec models from *gensim* library provide a word mover's distance implementation and is used here.

Listing 5.2: Python code for computing word mover's distance method between two sentences

```
def wmd(sent1, sent2, model, use_stoplist=False):
    tokens1 = [token.lower() for token in nltk.word_tokenize(sent1)]
    tokens2 = [token.lower() for token in nltk.word_tokenize(sent2)]
    stop_words = set(nltk.corpus.stopwords.words("english"))
    if use_stoplist:
        tokens1 = [token for token in tokens1 if token not in stop_words]
        tokens2 = [token for token in tokens2 if token not in stop_words]

    tokens1 = [token for token in tokens1 if token in model]
    tokens2 = [token for token in tokens2 if token in model]

    return (-model.wmdistance(tokens1, tokens2))
```

5.1.3 Smooth Inverse Frequency (SIF)

In case of cosine similarity, computing the average of the word embeddings in a sentence tends to give too much weight to words that are semantically irrelevant, such as but, just etc. Thus, Smooth Inverse Frequency (SIF) is used to modify the cosine similarity method presented in listing 5.1. The modified method is presented in listing 5.3. Unlike the previous two methods, *sif_similarity* takes all the sentences from two documents as parameters. Like tf-idf, SIF takes the weighted average of the word embeddings in the sentence. Every word embedding is weighted by $\frac{a}{(a+p(w))}$, where a is a parameter that is typically set to 0.001 and $p(w)$ is the estimated frequency of the word in a reference corpus calculated using a token frequency dictionary, provided by parameter *freqs*. Next, the principal component of the resulting embeddings for the set of sentences is computed. It then subtracts from these sentence embeddings their projections on their first principal component, which remove variations related to frequency and syntax that is semantically less relevant. The method returns a matrix of similarity measures for each sentence in one document to each sentence of the other document.

Listing 5.3: Python code for computing cosine similarity with SIF

```
def remove_first_principal_component(X):
    svd = TruncatedSVD(n_components=1, n_iter=7, random_state=0)
    svd.fit(X)
    pc = svd.components_
```

```
    XX = X - X.dot(pc.transpose()) * pc
    return XX


def sif_similarty(sentences1, sentences2, model, freqs={}, a=0.001):
    total_freq = sum(freqs.values())
    stop_words = set(nltk.corpus.stopwords.words("english"))
    embeddings = []

    for (sent1, sent2) in zip(sentences1, sentences2):

        tokens1 = [token.lower() for token in nltk.word_tokenize(sent1)]
        tokens2 = [token.lower() for token in nltk.word_tokenize(sent2)]


        tokens1 = [token for token in tokens1 if token not in stop_words]
        tokens2 = [token for token in tokens2 if token not in stop_words]

        tokens1 = [token for token in tokens1 if token in model]
        tokens2 = [token for token in tokens2 if token in model]

        weights1 = [a/(a+freqs.get(token,0)/total_freq) for token in tokens1]
        weights2 = [a/(a+freqs.get(token,0)/total_freq) for token in tokens2]

        embedding1 = np.average([model[token]
                                 for token in tokens1], axis=0, weights=weights1)
        embedding2 = np.average([model[token]
                                 for token in tokens2], axis=0, weights=weights2)

        embeddings.append(embedding1)
        embeddings.append(embedding2)

    embeddings = remove_first_principal_component(np.array(embeddings))
    sims = [cosine_similarity(embeddings[idx*2].reshape(1, -1),
                              embeddings[idx*2+1].reshape(1, -1))[0][0]
            for idx in range(int(len(embeddings)/2))]

```

5.2 Evaluation of Sentence Matching Techniques

As mentioned before, there are two sentence embedding methods that are to be evaluated in combination with three similarity measure methods. Instead of using a testing set from a generic corpora for evaluation, we decided to use the privacy policies from Facebook. This is done to decide on the method that gives the best results when it comes to privacy policy specific terms. Following sub-sections are dedicated to each combination of a embedding method and similarity measure. Facebook privacy policies from September 2016 and October 2018 were chosen for this evaluation of matching on sentence level. A pair of policies (A, B) for comparison is formed by a revision A of the policy, and a revision B of the policy, where B was released chronologically right after A. After manually going through all such pairs of policies of Facebook, pair (September 2016, October 2018) was considered the best choice for the evaluation of sentence matching. This pair had relevant changes between semantically same sentences from both the documents, but the overall changes were not so extreme as to make the whole matching process irrelevant. A ground truth was created from this pair, by manually pairing up the semantically same sentences from the two policies. Analysis and evaluations were done by calculating similarity measure of each sentence in September 2016 to every sentence in October 2018 and using the ground truth, similarity measure should have the highest value for the correct pair of matched sentences. This value should also have a high margin of difference from similarity measures for incorrect sentence pairs, which will contribute in minimizing mismatches.

5.2.1 Word2Vec with Cosine Similarity

Cosine similarity using word2vec model for embedding has 4 variations: 1) without eliminating stopwords and without token frequencies, 2) eliminating stopwords and without token frequencies, 3) not eliminating stopwords and using token frequencies and 4) eliminating stopwords and using token frequencies. Each of the four variations are manually evaluated using September 2016 and October 2018 policies of Facebook. We use the sentences:

> "We share information globally , both internally within the Facebook Companies, and externally with our partners and with those you connect and share with around the world in accordance with this policy"

from October 2018, whose corresponding match in September 2016 is

> "How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy."

to serve as an example pair; results and observations from each variation for this example comparison are discussed below.

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	0.866997
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	0.837886
We share information we have about you within the family of companies that are part of Facebook.	0.833949
Sharing With Third-Party Partners and Customers, We work with third party companies who help us provide and improve our Services or who use advertising or related products, which makes it possible to operate our companies and provide free services to people around the world.	0.829297
As you review our policy, keep in mind that it applies to all Facebook brands, products and services that do not have a separate privacy policy or that link to this policy, which we call the "Facebook Services" or "Services."	0.827557

Table 5.1: Top-five highest matching sentences to an example sentence using word2vec and cosine similarity with parameters use_stoplist = False and doc_freqs = None

Highest five cosine similarity measure of our chosen sentence from the October 2018 policy without eliminating stop words and not using frequencies for normalization (variation 1) is shown in table 5.1. This variation was successful in assigning the highest measure to the correct sentence pair, but the margin of difference with similarity measures of incorrect sentence pairs was too low. This low margin could lead to errors in matching.

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	0.811657
We share information we have about you within the family of companies that are part of Facebook.	0.716470
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	0.709647
Sharing With Third-Party Partners and Customers, We work with third party companies who help us provide and improve our Services or who use advertising or related products, which makes it possible to operate our companies and provide free services to people around the world.	0.700438
As you review our policy, keep in mind that it applies to all Facebook brands, products and services that do not have a separate privacy policy or that link to this policy, which we call the “Facebook Services” or “Services”.	0.667572

Table 5.2: Top-five highest matching sentences to an example sentence using word2vec and cosine similarity with parameters use_stoplist = True and doc_freqs = None

The second variation eliminated the stop words and did not use the token frequencies for normalization. Table 5.2 shows the highest five similarity measures for the selected example with this variation. The second variation showed much better results in terms of higher margin in similarity measure between correct sentence pairs and others.

Matched Sentence	Similarity Measure
How our global services operate,Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	0.858884
Sharing With Third-Party Partners and Customers, We work with third party companies who help us provide and improve our Services or who use advertising or related products, which makes it possible to operate our companies and provide free services to people around the world.	0.719372
In some cases, people you share and communicate with may download or re-share this content with others on and off our Services.	0.715968
As you review our policy, keep in mind that it applies to all Facebook brands, products and services that do not have a separate privacy policy or that link to this policy, which we call the “Facebook Services” or “Services.”	0.707121
We may provide these partners with information about the reach and effectiveness of their advertising without providing information that personally identifies you, or if we have aggregated the information so that it does not personally identify you.	0.702254

Table 5.3: Top-five highest matching sentences to an example sentence using word2vec and cosine similarity with parameters use_stoplist = False and doc_freqs = doc_frequencies

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	0.825778
In some cases, people you share and communicate with may download or re-share this content with others on and off our Services.	0.642143
Sharing With Third-Party Partners and Customers, We work with third party companies who help us provide and improve our Services or who use advertising or related products, which makes it possible to operate our companies and provide free services to people around the world.	0.637543
As you review our policy, keep in mind that it applies to all Facebook brands, products and services that do not have a separate privacy policy or that link to this policy, which we call the “Facebook Services” or “Services”.	0.634121
Public information is any information you share with a public audience, as well as information in your Public Profile, or content you share on a Facebook Page or another public forum.	0.624157

Table 5.4: Top-five highest matching sentences to an example sentence using word2vec and cosine similarity with parameters use_stoplist = True and doc_freqs = doc_frequencies

The highest five results from the third variation (without eliminating stop words but using the token frequencies) are shown in table 5.3. The margin of difference was higher than in the first and second variations. The last variation with both stop words eliminated and normalization with a frequency baseline showed the highest

margin of difference when compared to results from other variations, shown in table 5.4. The same observation was made when the four variations were tested with different sentences from October 2018. Thus, eliminating stop words and normalizing the embedding from word2vec with token frequencies produced the best results, when using word2vec and cosine similarity.

5.2.2 GloVe with Cosine Similarity

Matched Sentence	Similarity Measure
How our global services operate,Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	0.957919
Sharing With Third-Party Partners and Customers, We work with third party companies who help us provide and improve our Services or who use advertising or related products, which makes it possible to operate our companies and provide free services to people around the world.	0.945504
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	0.941013
We receive information about you and your activities on and off Facebook from third-party partners, such as information from a partner when we jointly offer services or from an advertiser about your experiences or interactions with them.	0.936749
We may provide these partners with information about the reach and effectiveness of their advertising without providing information that personally identifies you, or if we have aggregated the information so that it does not personally identify you.	0.935967

Table 5.5: Top-five highest matching sentences to an example sentence using glove and cosine similarity with parameters use_stoplist = False and doc_freqs = None

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	0.924916
Sharing With Third-Party Partners and Customers, We work with third party companies who help us provide and improve our Services or who use advertising or related products, which makes it possible to operate our companies and provide free services to people around the world.	0.882458
We may provide these partners with information about the reach and effectiveness of their advertising without providing information that personally identifies you, or if we have aggregated the information so that it does not personally identify you.	0.874468
We may associate the information we collect from your different devices, which helps us provide consistent Services across your devices.	0.864663
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	0.864304

Table 5.6: Top-five highest matching sentences to an example sentence using glove and cosine similarity with parameters use_stoplist = True and doc_freqs = None

Similar to the preceding section, there are four variations for GloVe with cosine similarity as well. These variations are : 1) without eliminating stopwords and without token frequencies, 2) eliminating stopwords and without token frequencies, 3) not eliminating stopwords and using token frequencies and 4) eliminating stopwords and using token frequencies. The same text from October 2018 is chosen as an example to explain the evaluation process:

> "We share information globally , both internally within the Facebook Companies , and externally with our partners and with those you connect and share with around the world in accordance with this policy".

Matched Sentence	Similarity Measure
We transfer information to vendors, service providers, and other partners who globally support our business, such as providing technical infrastructure services, analyzing how our Services are used, measuring the effectiveness of ads and services, providing customer service, facilitating payments, or conducting academic research and surveys.	0.902658
We may provide these partners with information about the reach and effectiveness of their advertising without providing information that personally identifies you, or if we have aggregated the information so that it does not personally identify you.	0.901763
Information we receive about you, including financial transaction data related to purchases made with Facebook, may be accessed, processed and retained for an extended period of time when it is the subject of a legal request or obligation, governmental investigation, or investigations concerning possible violations of our terms or policies, or otherwise to prevent harm.	0.896281
We may associate the information we collect from your different devices, which helps us provide consistent Services across your devices.	0.894429
Information collected by these apps, websites or integrated services is subject to their own terms and policies.	0.892844

Table 5.7: Top-five highest matching sentences to an example sentence using glove and cosine similarity with parameters use_stoplist = False and doc_freqs = doc_frequencies

Matched Sentence	Similarity Measure
We transfer information to vendors, service providers, and other partners who globally support our business, such as providing technical infrastructure services, analyzing how our Services are used, measuring the effectiveness of ads and services, providing customer service, facilitating payments, or conducting academic research and surveys.	0.894559
Information we receive about you, including financial transaction data related to purchases made with Facebook, may be accessed, processed and retained for an extended period of time when it is the subject of a legal request or obligation, governmental investigation, or investigations concerning possible violations of our terms or policies, or otherwise to prevent harm.	0.882535
We may provide these partners with information about the reach and effectiveness of their advertising without providing information that personally identifies you, or if we have aggregated the information so that it does not personally identify you.	0.872498
Public information is any information you share with a public audience, as well as information in your Public Profile, or content you share on a Facebook Page or another public forum.	0.868693
Information collected by these apps, websites or integrated services is subject to their own terms and policies.	0.866990

Table 5.8: Top-five highest matching sentences to an example sentence using glove and cosine similarity with parameters use_stoplist = True and doc_freqs = doc_frequencies

The first variation with neither eliminating stop words nor normalizing the weights with token frequency, assigned the highest similarity measure to correct pairs but the margin of difference from other incorrect pairs is very low. This low margin often lead to mismatch of sentences when tested with other sentences. This can be seen in table 5.5, where highest five measures and the corresponding matches are shown. If only stop words are eliminated (second variation), this margin of difference increases but not significantly, as seen in table 5.6.

Normalizing the word embeddings obtained from glove with or without stop words produced even poorer results, as can be seen in table 5.7 and 5.8, where the method failed to even assign the highest value to the correct sentence pairs and led to mismatch. Overall results from this evaluation showed that, when using cosine similarity, word2vec is a better choice for sentence matching process. This observation was further reinforced when glove with cosine similarity was tested with other sentences from October 2018 policy. Thus, this method was eliminated from the list of potential matching methods immediately.

5.2.3 Word2vec with SIF

As explained in section 5.1.3, to remove variations related to frequency and syntax that is less relevant semantically, SIF was applied on word2vec embedding with cosine similarity. Word2vec was selected as it outperformed glove (discussed in previous section). Policies from September 2016 and October 2018 are given as input to the method in listing 5.3. Table 5.9 shows five of the matched pairs formed when each sentence of the Facebook privacy policy of September 2016 was matched with a sentence from Facebook privacy policy of October 2018. The number of mismatches were too high, most pairs did not have any semantic similarity at all. Overall precision was 0.513 with this method, which was worse than the results we obtained with a simple cosine similarity with word2vec.

Text	Matched Text
This policy describes what information we collect and how it is used and shared	New owner
You can find additional tools and information at Privacy Basics	New owner
As you review our policy, keep in mind that it applies to all Facebook brands, products and services that do not have a separate privacy policy or that link to this policy, which we call the	Promote safety, integrity and security
“Facebook Services” or “Services”	New owner
What kinds of information do we collect? Depending on which Services you use, we collect different kinds of information from or about you	The types of information we collect depend on how you use our Products
Things you do and information you provide	Things you and others do and provide

Table 5.9: Word2vec cosine similarity (SIF normalized) sample matches

5.2.4 Word2vec and GloVe with Word Mover's Distance

Word Mover's distance (WMD) captures the semantic relationship between tokens in terms of distance in the vector space. For our evaluation the performance of word2-vec and glove are tested with two variations : one with stop words and other with stop words removed. Unlike cosine similarity, no token frequency is used in this evaluation to normalize the weights of vectors, as the vectors represent the position of the words in an N-dimensional space assigned by glove or word2vec. The closeness of two words in this space also determines the semantic closeness of the two words. Word mover's distance computes similarity by computing the distance between the tokens and hence, normalizing this weight will corrupt the semantic information embedded in the vector.

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	-1.519366
We share information we have about you within the family of companies that are part of Facebook.	-1.697253
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	-1.752882
We make this possible by sharing your information in the following ways: *People you share and communicate with.	-1.780076
In some cases, people you share and communicate with may download or re-share this content with others on and off our Services.	-1.854870

Table 5.10: Top-five highest matching sentences to an example sentence using word2vec and WMD with parameters, use_stoplist = False

We are using the same example sentence pairs:

> *"We share information globally , both internally within the Facebook Companies , and externally with our partners and with those you connect and share with around the world in accordance with this policy"*

and

> *"How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy."*

Note that: word mover's distance represents the distance between the tokens, which is a positive value. Smaller distance between two sentences implies more similarity. For ease of discussion and comparison with cosine similarity, the resultant word mover's distance is made negative by multiplying with -1 so that a higher value will imply higher similarity.

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	-1.676917
We share information we have about you within the family of companies that are part of Facebook.	-2.010831
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	-2.344054
Public information is any information you share with a public audience, as well as information in your Public Profile, or content you share on a Facebook Page or another public forum.	-2.465501
Sharing On Our Services, People use our Services to connect and share with others.	-2.516137

Table 5.11: Top-five highest matching sentences to an example sentence using word2vec and WMD with parameters, use_stoplist = True

Table 5.10 shows the results from word2vec with WMD for our example pair, without eliminating the stop words. Even though this method returned the highest similarity measure for most correct pairs, the margin of difference from other values was very low. Word mover's distance with glove and without eliminating the stop words had similar results, but the margin of difference of a correct pair similarity measure from the rest was observed to be a little higher, but not significant. Table 5.12 shows the similarity measures for the chosen example sentences while using glove with WMD, but without eliminating the stop words.

Eliminating the stop words produced much better results when using WMD, observed both for word2vec (table 5.11) and glove (table 5.13). But glove out performed word2vec in terms of higher margin and more consistency when the method was tested using other sentences from the October 2018 policy. Thus, glove with stop words removed was chosen as the ideal candidate for WMD.

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	-3.452347
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	-3.651142
We collect information about the people and groups you are connected to and how you interact with them, such as the people you communicate with the most or the groups you like to share with.	-3.709035
In some cases, people you share and communicate with may download or re-share this content with others on and off our Services.	-3.727923
We make this possible by sharing your information in the following ways: People you share and communicate with.	-3.769958

Table 5.12: Top-five highest matching sentences to an example sentence using glove and WMD with parameters, use_stoplist = False

Matched Sentence	Similarity Measure
How our global services operate, Facebook may share information internally within our family of companies or with third parties for purposes described in this policy.	-3.624838
We share information we have about you within the family of companies that are part of Facebook.	-4.320495
We receive information about you from companies that are owned or operated by Facebook, in accordance with their terms and policies.	-4.358536
Public information is any information you share with a public audience, as well as information in your Public Profile, or content you share on a Facebook Page or another public forum.	-4.625255
For example, people may share a photo of you, mention or tag you at a location in a post, or share information about you that you shared with them.	-4.724469

Table 5.13: Top-five highest matching sentences to an example sentence using glove and WMD with parameters, use_stoplist = True

5.2.5 Final Method for Sentence Matching

After the preceding evaluation and analysis, there were two potential methods that could be used for one-to-one sentence matching from one policy to another: word2vec embedding with cosine similarity (stop words removed and normalized with frequency tokens) and glove embedding with word mover's distance (stop words removed). To decide on the final method, instead of choosing one sentence from the October 2018 Facebook policy at a time to compare with every other sentence in the September 2016 policy, both the documents were taken as input. All the sentences in October 2018 (n sentences) and September 2016 (m sentences) were taken to conduct a $n \times m$ comparison. The result was a list of $m = 92$ pairs of sentences where for every sentence of September 2016, a sentence from the Facebook privacy policy of October 2018 with the highest similarity measure is selected to form a matched pair. The resultant matched pairs from both the potential methods are then manually checked against the ground truth for September 2016 - October 2018 Facebook policy matches. Precision was selected as an evaluation metric for finalizing the method. For word2vec embedding with cosine similarity, only 47 sentence pairs were correct out of 92 pairs, with a precision of 0.51. But glove embedding with WMD had a precision of 0.73 with a total of 67 correct pairing. Thus, glove with word mover's distance was selected as the final method for sentence matching. Since this evaluation requires manually going through each pair, we decided not to compute precision of both the methods for each privacy policy pair.

5.2.6 Threshold for Similarity Measure

Matching sentences without having a lower bound on similarity measure leads to formation of incorrect pairs. For a sentence which should not have a match with any sentence in the newer version of the policy (no semantically equivalent sentence due to deletion or a major policy revision), the absence of a threshold in similarity measures can lead to matching of such sentences with a sentence from the next policy. To prevent this, a threshold had to be estimated such that if the highest similarity measure for that sentence is lower than this threshold, then the pair should not be considered as a valid match and the sentence should remain unpaired. Matching of sentences without a threshold leads to high recall at the cost of precision (due to increase in false positive). A precision-recall curve for varying thresholds was plotted using the September 2016 and October 2018 Facebook privacy policy pair and shown in figure 5.1.

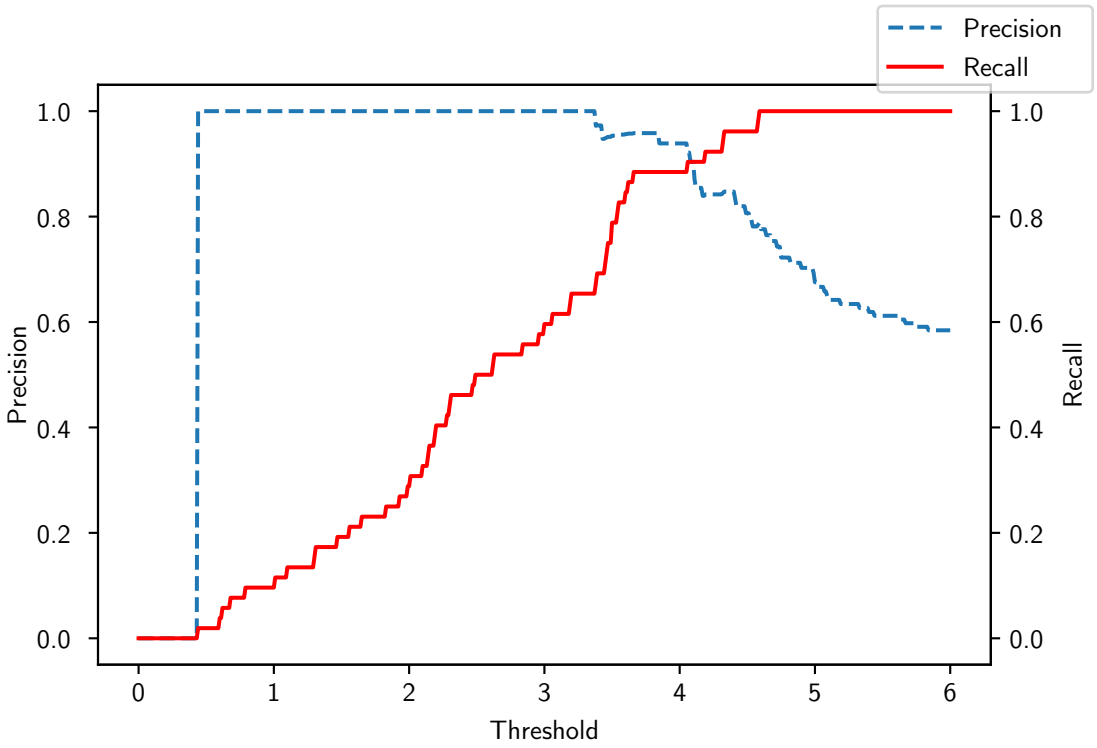

Figure 5.1: Precission - Recall - Threshold graph of sentence matching between Facebook privacy policies of September 2016 and October 2018 using glove with WMD

The intersection of precision and recall at a threshold around 4.4 was found to be an ideal point of acceptable precision and recall. Thus, 4.4 was selected as a threshold for similarity measures from word mover's distance to minimize invalid matches. For computing and plotting this precision-recall graph, a ground truth is necessary for all the policy pairs. Like the earlier evaluations only the September 2016 and October 2018 Facebook privacy policies are used.

5.2.7 Using a Sentence Classifier

The initial method for creating sentence pairs from two privacy policies was found to be computationally expensive. Suppose we have two privacy policies (P, N), where P is the older version of the policy with a sentences and N is the newer version of the policy following P with b sentences. The initial method to compute similarity measure for each sentence in P with each sentence of N and find the pair with the highest similarity, requires a total of $a \times b$ similarity measure computations.

To improve the computation speed and make the pairing process more efficient, we leverage our sentence classifier. We first categorized each sentence of both the documents using the classifier, then for each sentence of P, we computed the similarity measures with only those sentences in N that had the same expert-defined-policy-category as the sentence from P. If two sentences form a correct pair then both of them should belong to the same category as well. We used this idea to significantly improve the computation time.

Not only computation speed was improved, but a significant improvement in precision was also observed. For the same privacy policy pair of September 2016 and October 2018 Facebook policy, the precision improved from 0.73 to 0.83. Thus, sentence classification was incorporated into sentence matching. Sentence classification also presents the opportunity for more innovative solutions such as using sentence categories as keys to divide and distribute sentences among clusters in a parallel computing architecture for faster sentence matching. We did not implement this approach but it can serve as a potential future improvement over our current implementation.

5.3 Results and Discussion

We narrowed our choice to the method using word mover's distance and glove word embedding that uses a threshold value of 4.4 and sentence classification to improve precision and computation speed. For the performance evaluation of the developed method, it was first tested using all the collected Facebook policy pairs. The matched sentence pairs for each policy pair was then read individually and marked as correct or incorrect match based on semantic interpretation of the pairs. The whole process was intensive due to lack of any ground truth and high number of text pairs for each policy pair, all of which had to be read manually.

Base Policy	Next Policy	Total Pairs	Correct Pairs	Precision	Null Assigned
June 2005	February 2006	38	33	0.868	6
February 2006	May 2006	70	70	1	0

May 2006	October 2006	107	105	0.981	0
October 2006	May 2007	140	138	0.986	1
May 2007	September 2007	154	154	1	0
September 2007	December 2007	154	153	0.993	0
December 2007	November 2008	159	159	1	0
November 2008	December 2009	160	102	0.637	28
December 2009	October 2010	267	227	0.85	6
October 2010	December 2010	298	290	0.976	0
December 2010	January 2011	301	301	1	0
January 2011	September 2011	301	165	0.548	63
September 2011T	December 2012	300	285	0.95	4
December 2012	November 2013	379	356	0.939	5

November 2013	January 2015	367	193	0.525	118
January 2015	September 2016	120	120	1	2
September 2016	October 2018	92	76	0.826	11

Table 5.14: Facebook sentence matching across different policy revisions

Table 5.14 shows the result from the evaluation. Column "Base Policy" is the version of the policy that serves as the basis during sentence matching. "Next Policy" column stores the following version of the policy. The number of total matched sentence pairs will always be equal to the total number of sentences in the "Base Policy", as the pairs are formed by finding a suitable sentence in the next version. Column "Total Pairs" refers to the total number of pairs formed when "Base Policy" and "Next Policy" is given as input. Again, this value is same as the number of sentences in the "Base Policy". "Correct Pairs" stores the number of pairs that are correctly matched among "Total Pairs". "Precision" of the matching is simply the ratio "Correct Pairs" : "Total Pairs". In correctly matched pairs, there might be sentences of "Base Policy" which are assigned nothing (empty string or null) due to our use of the threshold. "Null Pairs" shows the number of pairs with a null assignment.

There are some interesting observations from this evaluation. Most policy pairs had their sentences matched with very good precision as can be seen in table 5.14. Taking a sentence pair example from one of these policy pairs, the method matched sentences:

> *"This includes your payment information , such as your credit or debit card number and other card information , and other account and authentication information , as well as billing , shipping and contact details "*

with

> *"This includes payment information , such as your credit or debit card number and other card information ; other account and authentication information ; and billing , shipping and contact details .",*

shows that the method can find and pair sentence, that are present in both the policies without any difficulty. Another example pair such as

> *"Depending on which Services you use , we collect different kinds of information from or about you "*

and

> *"The types of information we collect depend on how you use our Products ."*

The method's capability in handling slightly more complex cases where the structure of both the sentences have changed slightly with addition and deletions of some tokens ("Services" was replaced with "Products") can be seen. The method still managed to pair these two semantically same sentences. The same also holds for complex sentence pairs with a lot of changes between them but still having the same semantics. For example, in the pair:

> *"In addition , when you download or use such third - party services , they can access your Public Profile , which includes your username or user ID , your age range and country / language , your list of friends , as well as any information that you share with them "*

and

> *"Also , when you download or use such third - party services , they can access your public profile on Facebook , and any information that you share with them ."*

There has been a significant deletion of words between the two sentences even though they mean the same thing, the method successfully matches them. Using a threshold has also prevented some false matches. For the sentence:

> *"This policy describes what information we collect and how it is used and shared ."*

There does not exist a sentence in the "Next Policy" having the same semantics, and our threshold made sure that this sentences doesn't get assigned to any random sentence without any semantic similarity. "Null Assigned" in table 5.14 shows the number

of such sentences without any corresponding sentence in the "Next Policy" and which are successfully not matched with any sentence due to the threshold.

As can be seen in table 5.14, for policy pair: November 2008 - December 2009, January 2011 - September 2011 and November 2013 - January 2015 of Facebook, the precision is significantly low. When each of these policies were examined carefully, it was found that the revision from one version to the next had some very significant structural changes. Most sentences for the "Base Policy" in these pairs did not have any semantically equivalent match in the "Next Policy". Ideally all the deleted sentences from "Base policy" should not have been assigned any sentence from the next policy. Our threshold managed to achieve that to some extent as can be seen from the high "Null Assigned" numbers for these pairs. But this was still not enough to prevent enough false matches for a better precision. Lowering the threshold would have reduced the false matches, but it would have prevented the method from matching semantically same sentences with significant changes, such as

> *"In addition , when you download or use such third - party services , they can access your Public Profile , which includes your username or user ID , your age range and country / language , your list of friends , as well as any information that you share with them "*

and

> *"Also , when you download or use such third - party services , they can access your public profile on Facebook , and any information that you share with them ."*

Also during our investigation into these policies with lower precision, it was noticed that the amount of change was so high that comparing them on a sentence level does not make sense to begin with. Since the purpose of sentence matching is to compare versions of policy and extract changes between them at a sentence level, a significant structural and semantic change between the versions make sentence matching a futile process.

We plotted the precision of all the versions of the Facebook policies in a chronological order to give a visual representation of such extreme structural and semantic change during each revision of the policy. Figure 5.2 shows the precision we obtain from matching sentences of "Base Policy" and "Next Policy".

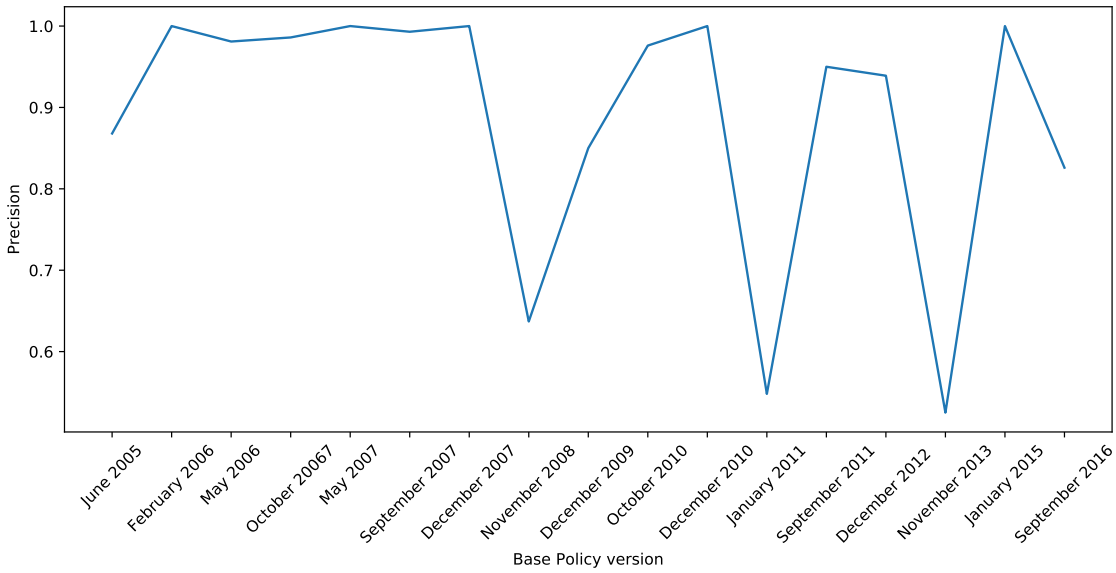

Figure 5.2: Precision of sentence matching across Facebook policies

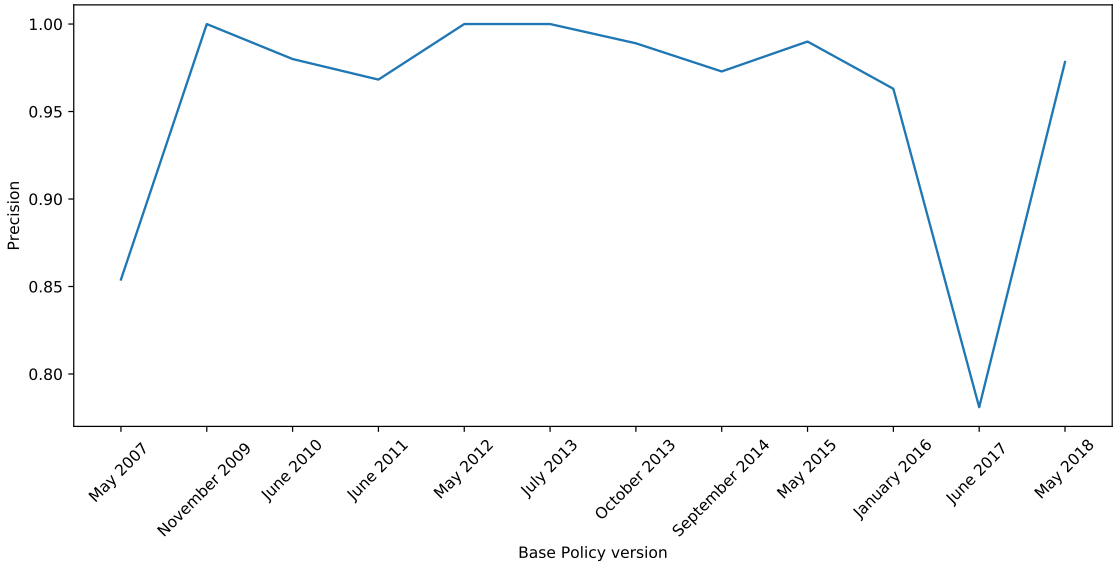

Figure 5.3: Precision of sentence matching across Twitter policies

The dips in the graph show major change in the following policy resulting in low precision. The same precision graph was plotted for the Twitter and WhatsApp policies, as shown in figure 5.3 and figure 5.4 respectively. Sentence matching in both Twitter and WhatsApp performed better than Facebook for all the versions of the policies. Even though during analysis and development we used only the Facebook privacy pol-

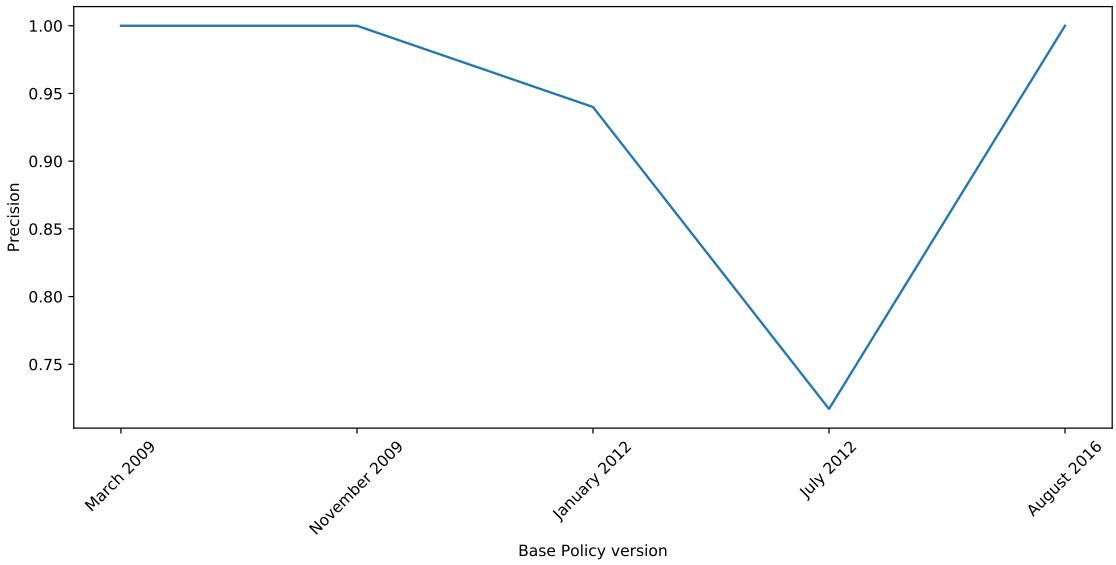

Figure 5.4: Precision of sentence matching across Whatsapp policies

icy versions September 2016 and October 2018, this did not lead to deficient sentence matching for other versions of Facebook privacy policies and also for privacy polices of other companies. The graph shown in figure 5.3 shows slightly significant changes for Twitter policies during revision May 2007 to November 2009 and June 2017 to May 2018; the method still matched sentences with a precision not below 0.80, showing very good performance. For the WhatsApp policies, during the revision from July 2012 to August 2016, the method had a precision of 0.717. The revision of WhatsApp privacy policy from the July 2012 to August 2016 version was done when Facebook purchased WhatsApp, and on careful study of both the policies, it was found that some major changes in the policy was introduced during this change of ownership. Despite that, a precision of 0.717 was observed, with 42 sentences assigned null matches showing that our decided threshold was able to prevent significant number of irrelevant matches. Another observation here, since our threshold was determined using only the Facebook privacy policy pair of September 2016 and October 2018, there is a possibility that the results discussed here could have been improved by using a different threshold. But the lack of ground truth and the extremely laborious nature of creating a proper ground truth, makes it impossible for us to determine a globally optimized threshold.

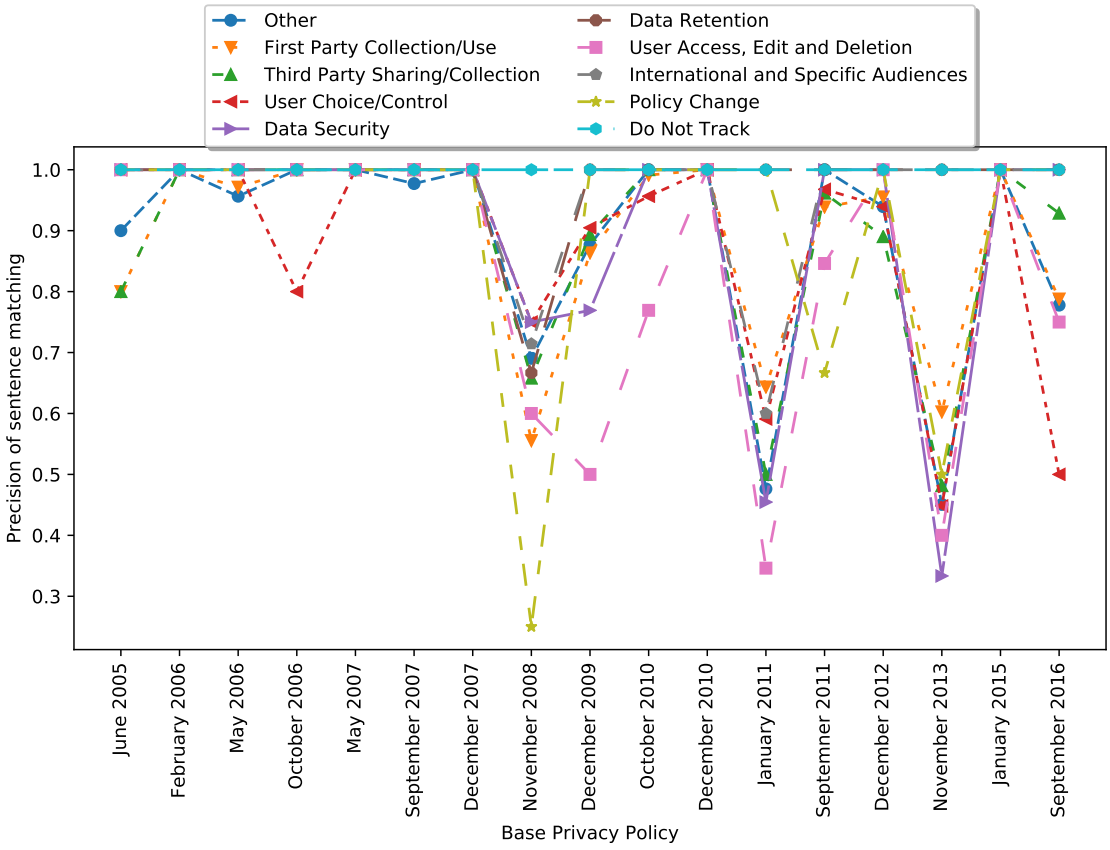

Figure 5.5: Precision of sentence matching across categories in Facebook policies

Similar to the analysis discussed towards the end of chapter 4, we wanted to see the performance of our sentence matcher for each category. Figure 5.5 shows the plot for precision in each category through all the Facebook policy pairs. As can be seen in the plot, the category of the sentence has no correlation with the precision of the sentence matching. In case of major changes, the drop in precision is lowest for categories of "Other", ''First Party Collection/Use" and "Third Party Sharing/Collection", but even that is not always true. The drop in precision for other categories is observed to be higher; the low number of sentences for these categories is a major factor behind this. Low number of sentences means even if the number of wrongly matched sentenced pairs is low, the precision gets more affected.

In summary, evaluation of the sentence matching technique implemented by us, showed efficient performance and proved to be an applicable technique. We also showed the prevention of false matches using a threshold value which improved the performance of sentence matching.

6. Change Detection

The previous chapter discusses matching sentences of a policy to a semantically similar sentence in the next version of the policy and using sentence classification to improve this matching process. The next step after matching two sentences is to detect the changes between the two matched sentences. The change detection process shall extract the user relevant changes between two matched sentences by identifying the additions and deletions between them. This identification will help notify users of informed changes in a more readable and comprehensible way, enabling them to understand the changes introduced in the newer revision of the policy with more ease.

6.1 Parse Tree

The first approach explored to detect the changes between the two sentences is to generate a parse tree of the two sentences and compare their structures to detect changes between them. We decided to identify changes in nouns and verbs describing the action, state, or occurrence of the said noun. Hence, during tokenization of the sentences, only noun or verb tokens are kept and tokens with other parts-of-speech (POS) tags are removed from the list (refer to section 2.2.1 for more details on tokenization). Sentences comprising of only nouns and verbs are then chunked into parse trees using the *nltk.chunk parser* and the chunk regular expression rule:

Entity:{(<VBP|VBZ|VB><NN|NNS|NNP><NN|NNS|NNP>)}*

Were *VB,VBP* and *VBZ* represents *nltk library* POS tags for verbs, and *NN* and *NNS* are POS tags for nouns and perfect nouns. *VB,VBP* and *VBZ* are tags for verb base, non third person singular present and third person singular present forms respectively.

The chunking returns a tree where the verb token and their corresponding subject noun tokens are grouped together in the same subtree, as leaf nodes of the subtree.

Each of these subtrees are labeled as entity and they represent the context of the noun tokens in the sentence. For sentences:

> S_1: *"Similarly, when you use Messenger or Instagram to communicate with people or businesses, those people and businesses can see the content you send"*

from Facebook privacy policy of October 2018 and

> S_2: *"Likewise, when you use Messenger, you also choose the people you send photos to or message"*

from Facebook privacy policy of September 2016, their respective parse trees are shown in figure 6.1 and figure 6.2 respectively.

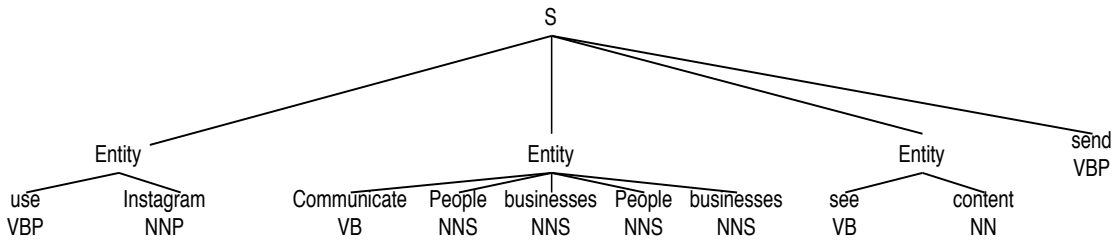

Figure 6.1: Parse tree for S_1

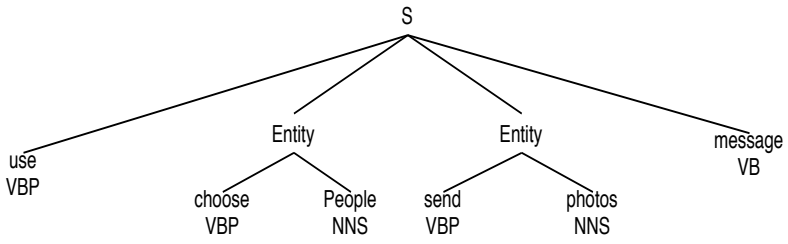

Figure 6.2: Parse tree for S_2

We tried using additions and deletions of entity subtrees in the parse trees of the matched sentences, and changes in the structure of entity subtree to extract changes between two sentences. But there are a few problems with this approach that made detecting changes difficult. The first problem identified was incorrect tagging of parts of speech for some tokens. For example, in both sentence, the word "Messenger" was

not identified as a noun, instead it was tagged as an adjective (*nltk* tag *JJR*), leading to incorrect entity subtree formation for "Messenger". Ideally the parse tree for S_1 should have had an entity subtree with nodes: "use", "Instagram" and "Messenger", and the second tree should have had an entity subtree with nodes "use" and "Messenger". Theoretically, the change could have been detected by comparing the two subtrees from both sentences; the incorrect tagging of "Messenger" prevented that. Another problem observed with using a chunk parser is that some of the dependency links get lost. Dependency links represent the relationship between two tokens in a sentence. For example, in S_1, the dependency link between the verb "send" and it's subject "content" was lost due to the unilateral nature of the chunking process. Hence, "send" and "content" are not in the same entity subtree.

Even though this approach failed, the analysis of it helped in formulating the final method for detecting changes. We observed that user-relevant changes can be detected by identifying new or deleted nouns tokens, and using the dependency links of these tokens to identify the context verb. To implement this approach, we decided to use a dependency tree.

6.2 Dependency Tree

Dependency tree is a parse tree based on the dependency grammar. Dependency grammar captures the relationship between words or tokens in a sentence. The notion behind this is that every word is connected to each other by a direct or an indirect link. A verb token of the sentence is taken to be the structural center of the dependency tree and other words are either directly or indirectly connected to this verb token in terms of the directed links, which are called dependencies. The overall structure of the tree is determined by the relation between the head token (center verb or root word) and it's dependent tokens (other words in the sentence) connected by these links.

Consider the matched sentences

S_3: *"This information is gathered for all users to the Web Site"*

from Facebook privacy policy of June 2005 and

S_4: *"This information is gathered for all Facebook visitors"*

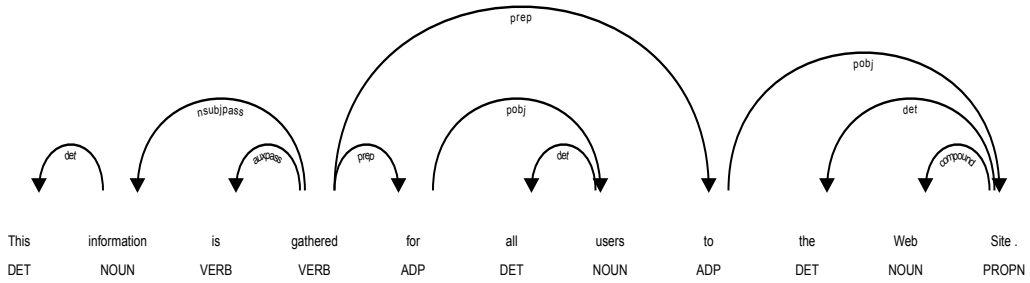

Figure 6.3: Dependency tree for S_3

from Facebook privacy policy of February 2006. Their respective dependency trees parsed using *spacy* package and *en_core_web_sm* are shown in figure 6.3 and figure 6.4 respectively.

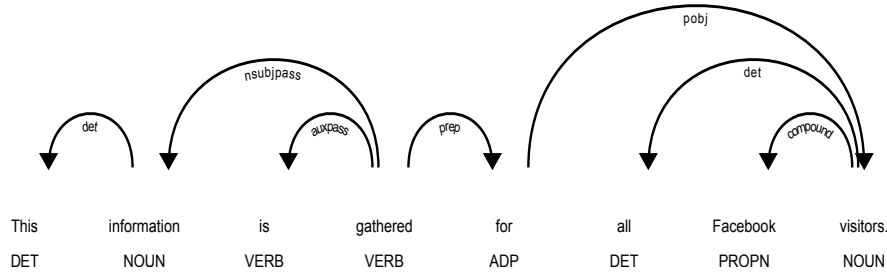

Figure 6.4: Dependency tree for S_4

Model *en_core_web_sm* [1] is an English multitask CNN provided by *spacy*. It provides general-purpose pre-trained models to predict named entities, part-of-speech tags and syntactic dependencies. As seen in figure 6.3 and figure 6.4, dependency based parse trees provides a better parts-of-speech tagging for the tokens using dependencies between between words rather than using a dictionary for POS tagging. It provides relationship labels between tokens as well which are used in our algorithm. In the figures 6.3 and 6.4, labels *DET, NOUN, VERB, ADP* and *PROPN* are *spacy* POS tags

[1] https:/spacy.io/models

for determiner, noun, verb, adpositon and proper noun respectively. Edge labels: *det* (determiner), *nsubjpass* (passive nominal subject), *auxpass* (auxiliary passive), *prep* (prepositional modifier), *pobj* (object of preposition) and *compound* (compound) represent the dependency relationships in the figures. The Python code used to extract the changes between matched sentences using their dependency tree is shown in the listing 6.1.

Listing 6.1: Python code for extracting changes between two sentences

```
def getTokenContext(tokenList):
    verbTokens = set()
    for token in tokenList:
        tokenTemp = token
        while (tokenTemp.dep_ != 'ROOT'):
            if tokenTemp.pos_ == 'VERB':
                verbTokens.add(tokenTemp)
                break
            tokenTemp = tokenTemp.head
    return verbTokens
def getChanges(text1, text2):
    dict1 = {}
    dict2 = {}
    tree1 = dependencyTree(text1, None)
    tree2 = dependencyTree(text2, None)
    for token in tree1:
        if token.pos_ == 'NOUN' or token.pos_ == 'PROPN':
            tokenTemp = token
            keyString = token.text
            while(tokenTemp.dep_ == 'compound' or tokenTemp.dep == 'amod'):
                keyString = keyString + ' ' + token.head.text
                tokenTemp = tokenTemp.head
                if tokenTemp.text == token.text:
                    break
            dict1[keyString] = token
    for token in tree2:
        if token.pos_ == 'NOUN' or token.pos_ == 'PROPN':
            tokenTemp = token
```

```
29              keyString = token.text
30              while(tokenTemp.dep_ == 'compound' or tokenTemp.dep == 'amod'):
31                  keyString = keyString + ' ' + token.head.text
32                  if tokenTemp.text == token.text:
33                      break
34              dict2[keyString] = token
35      deletionNounTokens = [v for k,v in dict1.items() if k not in dict2]
36      additionNounTokens = [v for k,v in dict2.items() if k not in dict1]
37      verbTokens1 = getTokenContext(deletionNounTokens)
38      verbTokens2 = getTokenContext(additionNounTokens)
```

The method *getChanges* takes the two strings (*text1* and *text2*) as input and generate a dependency tree for each of them. Then, the trees are traversed to identify nouns and proper nouns, and other tokens having compound or adjective modifier relationship with the identified noun/proper noun tokens. Two dictionaries (*dict1* and *dict2*) are formed for both the sentences using the identified tokens along with their adjective modifier and compound tokens as the key and the tokens as the value (lines 12-34 in listing 6.1). Subtraction between these two dictionaries using the keys gives the deleted (*dict1-dict2*) and added (*dict2-dict1*) noun tokens between two sentences. The purpose of forming keys in this manner is to eliminate false detection due to different arrangement of words but implying the same meaning. Using only noun tokens without their adjective modifiers can lead to missing out of some of the important changes introduced by it. After the new or deleted tokens are identified, method *getTokenContext* is used to identify the context verbs for a token by traversing from the tokens in the list (*verbTokens1* and *verbTokens2*) to the *root* of the tree using the dependency links. The dependency links ensure that there exists a path to the action verb of the token, if it exists, thus, identifying the action or state of the token.

Using the above methods, changes between two matched sentences are extracted in terms of token lists that are then used to appropriately highlight the changes between the sentences. This provides a comprehensible document that can help users understand key changes by looking at any two matched sentences.

6.3 Results and Discussion

We begin our evaluation by inputting matched sentence pairs from Facebook policies, consisting of different complexities, into our change detection method and analyzing the returned results. For exactly same sentences in two versions of the policy, such as

"What kinds of information do we collect ?",

the method did not highlight any part of the sentence. Redundant detection is prevented by using the similarity measure between the two sentences. For a similarity measure of 0, the method does not execute the detection algorithm, thus preventing redundancies. For slightly changed matched sentences, such as

"Depending on which Services you use , we collect different kinds of information from or about you."

and

"The types of information we collect depend on how you use our Products .",

the method highlighted the following changes (marked with an underline in the example and context verbs are italicized):

"Depending on which Services you *use* , we collect different kinds of information from or about you."	'The types of information we collect depend on how you *use* our Products ."

Our method successfully detected the change from "Services" to "Products". Replacement of "different kinds" with "types" is also highlighted. The token "use", which defines the action on the changed tokens is also detected. Note that using relationship links between tokens ensured that "different" and "kinds" are treated as a single entity. We mark detection efficiency of this form as "All", where all the relevant changes are highlighted. The method detected all the changes for some complex matches, which are also classified under the "All" category. For example:

"We collect information about the people and groups you are connected to and how you interact with them , such as the people you communicate with the most or the groups you like to share with ."	"We collect information about the people, Pages, accounts, hashtags and groups you are connected to and how you *interact* with them across our Products, such as people you communicate with the most or groups you *are* part of ."

Another interesting example, for the pair:

"You can find additional tools and information at Privacy Basics ."

and

"You can find additional tools and information in the Facebook Settings and Instagram Settings .",

we observed:

"You can find additional tools and information at Privacy Basics ."	"You can find additional tools and information in the Facebook Settings and Instagram Settings ."

Even though the method highlighted the relevant changes, it should have also highlighted "Settings" after "Facebook" in the second sentence. The relationship between "Facebook" and "Settings" was not captured in the dependency tree and hence, the two words were not treated as a single entity. We observed similar failure to detect some of the compound nouns in other examples as well. Detection which missed minute changes are marked as "Almost all". For some sentences, the method missed some of the relevant changes. Consider the example pair:

"When you comment on another person's post or like their content on Facebook, that person decides the audience who can see your comment or like."

and

" Also, when you comment on someone else's post or react to their content, your comment or reaction is visible to anyone who can see the other person's content, and that person can change the audience later."

We observed:

"When you *comment* on another person's post or like their content on Facebook, that person decides the audience who can see your comment or like."	"Also, when you *comment* on someone else's post or react to their content, your comment or reaction is visible to anyone who can see the other person's content, and that person can change the audience later."

In this example, the method did not completely detect the change from "person's" to "someone else's". Also the subtle change in the language used for audience selection of the post was missed by our method. This is due to the fact that we are using changes in noun to derive other changes between the sentences, which sometimes fail to detect some of the subtle changes. We mark such a detection as "Partial".
Sometimes our method does not detect any change. We mark a failed detection as "None". The failed detection in changes are generally observed to involve numbers or URLs. For the matched sentences from WhatsApp:

"If WhatsApp learns that personally identifiable information of persons under 18 years of age has been collected on the WhatsApp Sites, then WhatsApp may deactivate the account and/or make the status submissions inaccessible."

and

"If WhatsApp learns that personally identifiable information of persons under 16 years of age has been collected on the WhatsApp Sites, then WhatsApp may deactivate the account and/or make the status submissions inaccessible.",

the only change is "18" to "16" in the sentence. The method fails to capture this as it is not tuned to detect changes in numerical values. Many times policies have reference to

section numbers or have links with numbers in them, changes in which are irrelevant to the user. Trying to capture numerical changes would have resulted in extracting these irrelevant changes as well. Thus, in changes involving numerical values, such as age or date, our method fails to capture them.

The last kind of detection we observed are marked as "Redundant". An example of a "Redundant" detection is:

"For example, people may share a photo of you , mention or tag you at a location in a post, or share information about you that you shared with them."	"For example, people can share a photo of you in a Story, mention or tag you at a location in a post , or share information about you in their posts or messages."

Here, the method captured the relevant addition of "Story". But it also made some false detection like "mention" and "posts". We have identified some instances where "Redundant" detection take place. Change in grammatical numbering of a word, presence of compound nouns, URLs and apostrophes sometimes result in redundancies. When a single sentence is split into two or more sentences in the next revision or vice-versa, our sentence matching method may return multiple pairs with the same sentence in it. Each of these pairs will be semantically incomplete and when such pairs are processed through the change detection method, redundant detection takes place.

Policy Pairs	Sentence Pairs	None	Partial	Redundant	Almost All	All
June 2005 February 2006	27	0	0	1	0	26
February 2006 May 2006	70	0	0	3	0	67
May 2006 October 2006	105	0	0	2	0	103
October 2006 May2007	137	1	0	5	4	127

May2007 September 2007	154	0	0	0	0	154
September 2007 December 2007	153	0	0	2	0	151
December 2007 November 2008	159	0	0	2	1	156
November 2008 December 2009	74	0	0	4	3	67
December 2009 October 2010	221	1	1	8	4	207
October 2010 December 2010	290	0	1	0	0	289
December 2010 January 2011	301	0	0	0	0	301
January 2011 September 2011	102	2	4	5	0	91
September 2011 December 2012	281	1	0	1	0	279
December 2012 November 2013	351	0	1	14	0	336
November 2013 January 2015	75	1	4	4	3	63
January 2015 September 2016	118	0	0	0	0	118

September 2016 October 2018	65	0	3	5	11	46

Table 6.1: Change detection across privacy policy pairs of Facebook

After analyzing multiple examples and identifying the categories (or level) of detection (“None”, “Partial”, “Almost all” and “All”), we manually went through each of the privacy policy pairs of Facebook and classified detected changes between the paired sentences into either of these categories. Table 6.1 shows the results from the evaluation of change detection across privacy policy pairs of Facebook. Column “Sentence Pairs” is the number of correctly matched sentence pairs. This number does not include the number of null assigned pairs. Columns “None”, “Partial”, “Redundant”, “Almost All” and “All” represents the observed detection levels for each pair. As can be seen, the change detection method captures “All” changes for most sentence pairs. We plotted the ratio of each detection category in change detection between the privacy policy pairs of Facebook. The overall precision of the sentence matching between the policy pairs are also plotted in the same graph.

The plot in figure 6.5 shows that precision of sentence matching between the policies do not affect the detection between the sentence pairs. The ratio of “All” detection is in the range of 0.8 to 1.0, except for the Facebook policy pair of September 2016 - October 2018. This also implies that for any correct pair of sentences, our method is most likely to extract all the relevant changes between them. The same graph was plotted for Twitter and WhatsApp privacy policies using the manual evaluation results of the change detection method, shown in figure 6.6 and figure 6.7 respectively. We observed similar results for policies from both Twitter and WhatsApp, where “All” changes are detected for most sentence pairs.

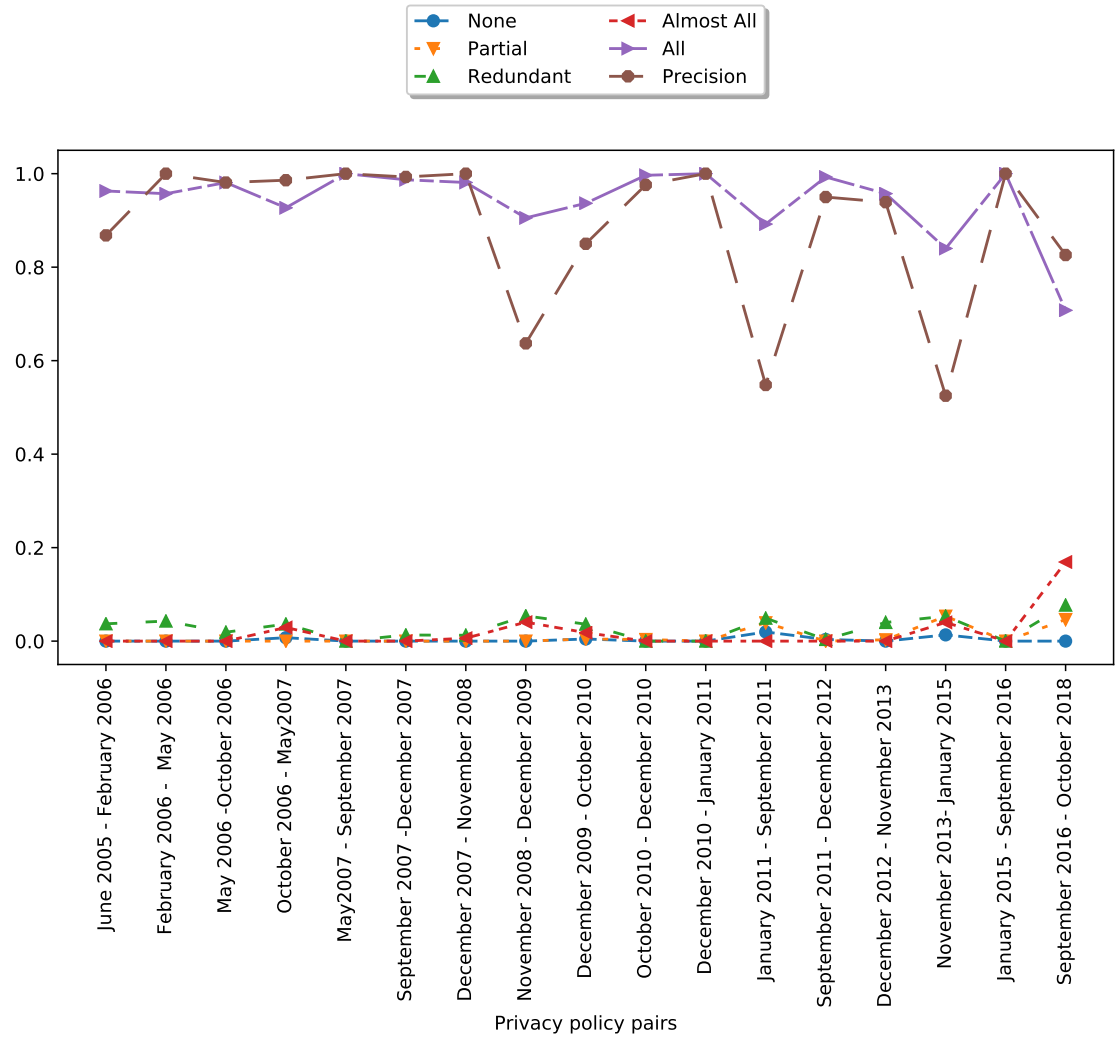

Figure 6.5: Precision of sentence matching and ratio of detection levels across Facebook policies

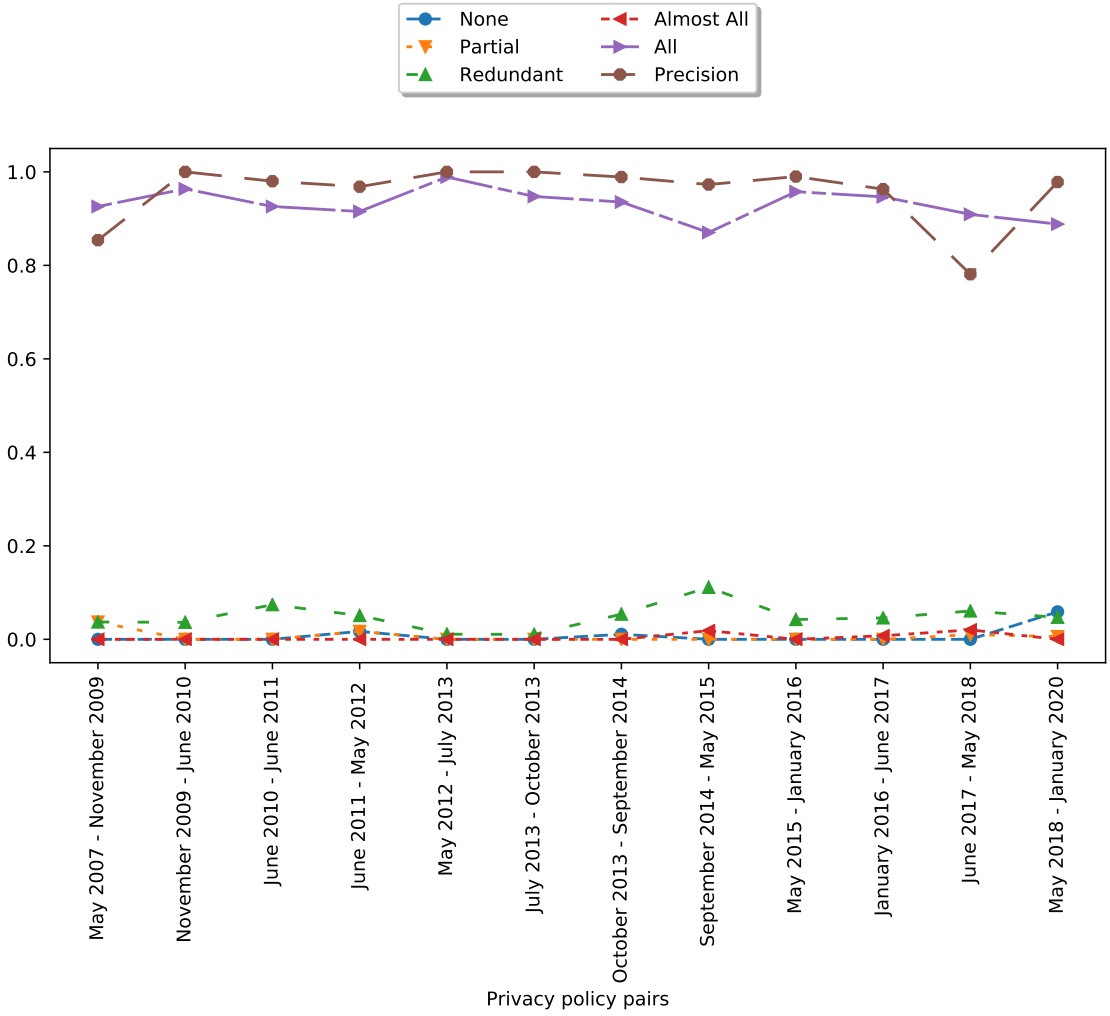

Figure 6.6: Precision of sentence matching and ratio of detection levels across Twitter policies

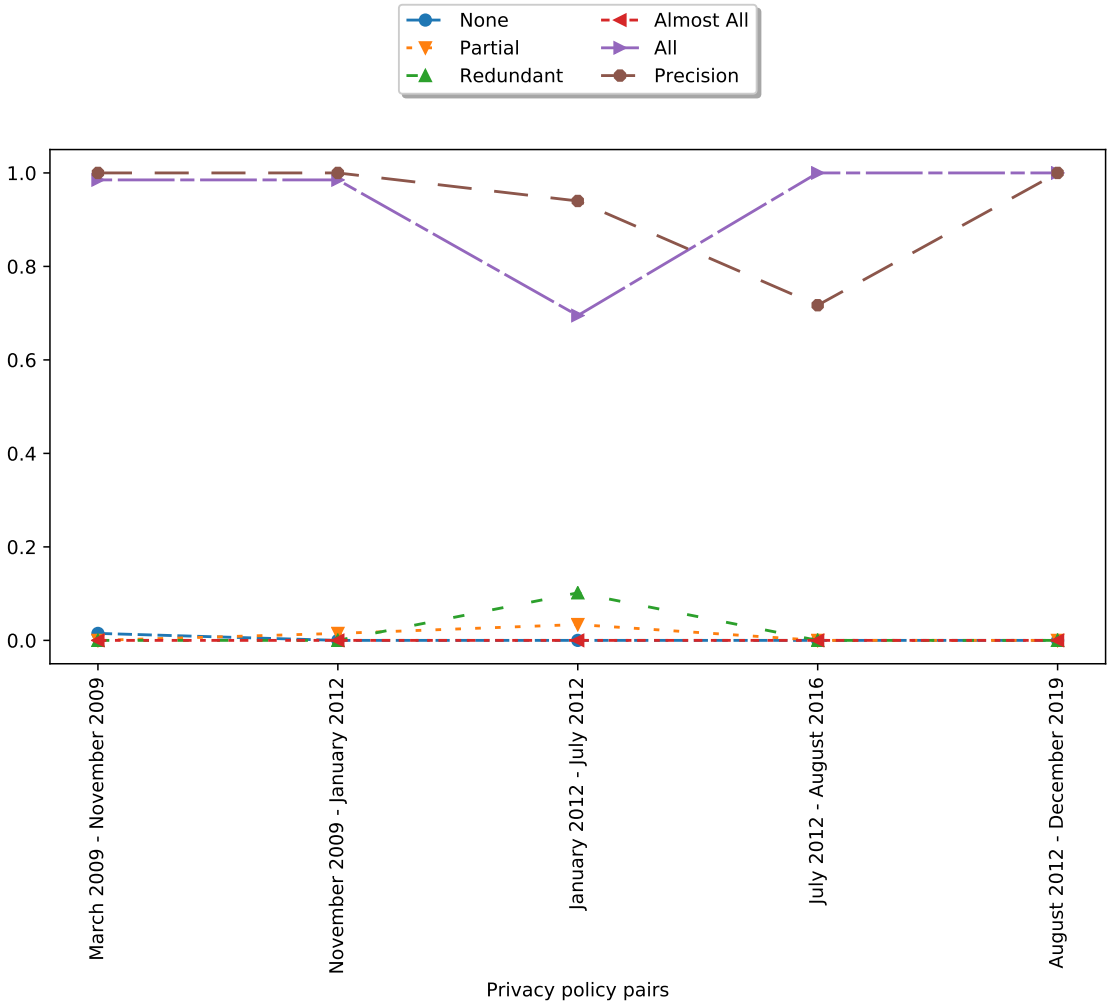

Figure 6.7: Precision of sentence matching and ratio of detection levels across WhatsApp policies

7. Extended Evaluation

Chapters 4, 5 and 6 discussed evaluation of our sentence classifier, sentence matching and change detection methods on policies of Facebook (18 policies), WhatsApp (6 policies) and Twitter (14 policies), which used ground truths of the respective policies. Additionally, we collected policies of Google (27 policies), LinkedIn (5 policies) and Snapchat (9 policies) totaling to 79 policies in our data set. We conducted evaluation and analysis of the performance of our methods and composition of the policies using this extended data set.

7.1 Execution Time

Execution time plays a vital role in the application of any proposed method. For each policy in the data set, we measured the execution time for sentence classification, sentence matching and change detection. We measured the time for each method 10 times for each policy, which was averaged to get the final estimate of the execution time. The estimated execution times for a policy are plotted against the number of sentences in that policy, shown in figure 7.1. The system used for the estimation has a 3.9-GHz, six-core Intel Core i7-8750H processor, 16GB of RAM, a 512GB M.2 SSD and an Nvidia GeForce GTX 1070 Max-Q GPU with 8GB of RAM.

We observed that sentence classification and change detection takes around 0.3 seconds on average. Low constant time execution for classification and detecting change between sentence pairs shows that both these methods are efficient and can be included in a real-time application for policy comparison. Sentence matching is observed as the bottle neck, computation time reaching as high as 10 minutes for a large policy pair. The $O(n^2)$ complexity of generating pairs makes it less efficient. Segregating sentences categorically slightly improved the performance. This is a one time task only needed when new policy versions appear.

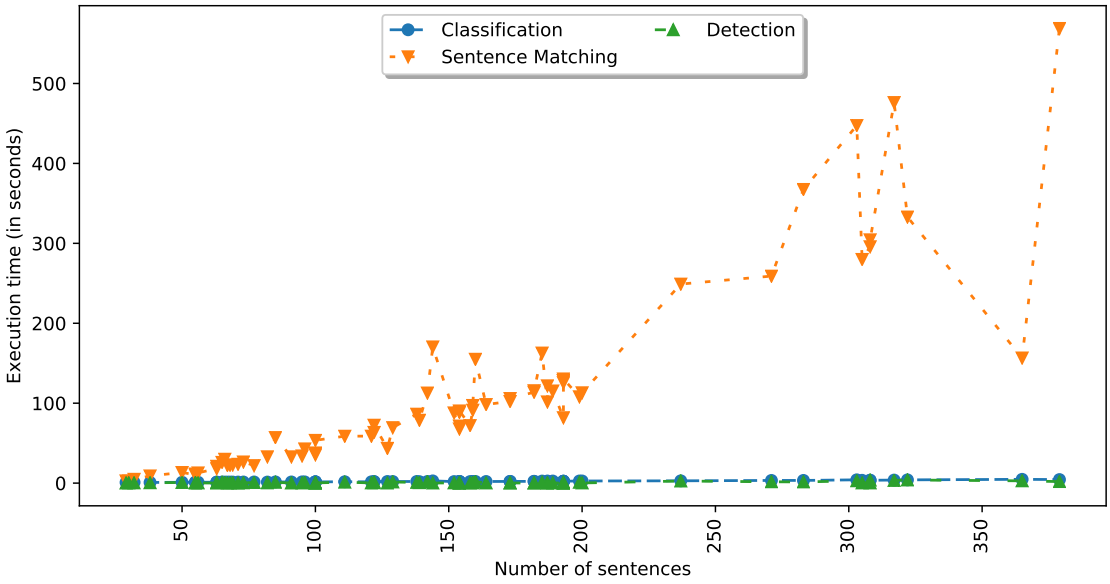

Figure 7.1: Execution time of sentence classifier, sentence matching and change detection method

7.2 Sentence Classification

Category	Percent composition (%)
Other	18.44
First Party Collection/Use	41.87
Third Party Sharing/Collection	19.16
User Choice/Control	6.56
Data Security	3.34
Data Retention	1.83
User Access, Edit and Deletion	5.07
International and Specific Audiences	1.63
Policy Change	2.07
Do Not Track	0.03

Table 7.1: Distribution of categories across collected policies

We categorized all the 12188 sentences in the 79 privacy policies using the sentence classifier. The category wise percentage composition of the data set is presented in table 7.1. The result shows most of the sentences belong to “First Party Collection/Use”, followed by “Third Party Sharing/Collection” and “Other”. The result is on par with the

statistics observed by Wilson et al. [37]. Collection, use and sharing of user data is a concern for most users. The primary purpose of privacy policies is to address that by notifying users of ways a party gathers, uses, discloses, and manages user data. High percentage of "First Party Collection/Use" and "Third Party Sharing/Collection" supports the primary purpose of the policies, but high number of "Other" category sentences also raises concern on the ambiguity of these documents. Low number of "User Choice/Control", "Do Not Track", "User Access, Edit and Deletion" and "Data Retention" cumulatively comprising of 13.49 percent of data set, shows how limited control users have over their own data. Even if the users are provided with the options of controlling the use and collection of data from them, policies do not provide detailed explanation or instructions to the users, which is another concern.

7.3 Sentence Matching

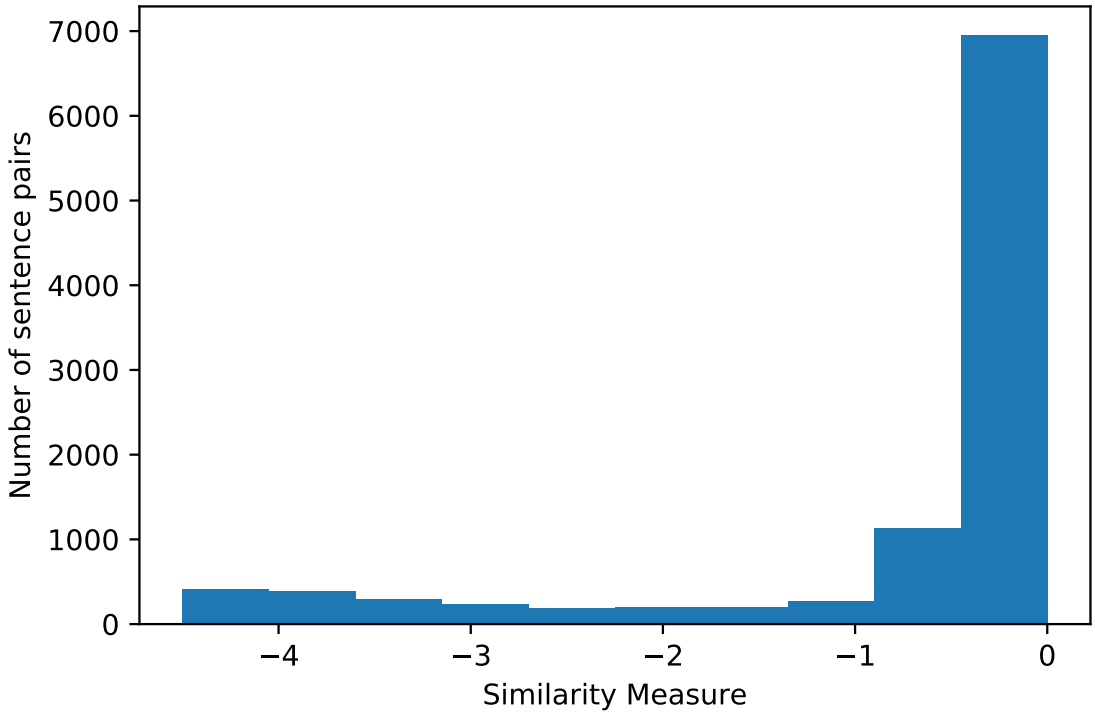

Figure 7.2: Similarity measure histogram

We plotted a histogram for the similarity measures of all the 10249 sentence pairs

generated for all the policy pairs in the data set, shown in figure 7.2. We have 9 bins for the similarity measures in the plot (0.0, -0.5), (-0.5, -1.0), (-1.0, -1.5), (-1.5, -2.0), (-2.0, -2.5), (-2.5, -3.0), (-3.0, -3.5), (-3.5, -4.0) and (-4.0, -4.5). Our threshold is -4.4; any sentence with highest similarity measure lower than -4.4 is null assigned and the range of similarity measure in the plot is set from -4.5 to 0. High number of pairs in the (0, -0.5) bin shows that most sentences are matched with very high confidence. Reading and checking the correctness of all the sentence pairs is a laborious task, but this plot gives an estimate of the confidence in our sentence matching method. High number of sentences pairs with higher similarity measures bolsters our confidence in the implemented method.

7.4 Change Detection

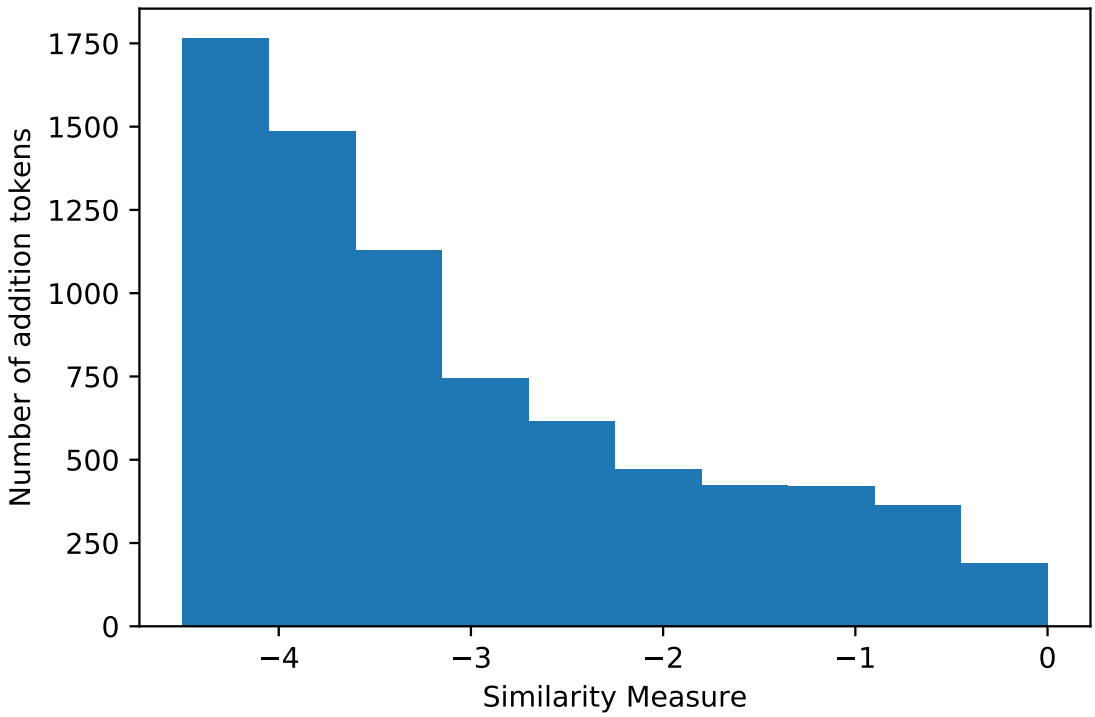

Figure 7.3: Distribution of addition tokens between sentence pairs

Using the change detection method, we collected the number of addition and deletion tokens between generated sentence pairs and divided them into bins using the

similarity measure of the sentence pairs. Distribution of addition and deletion tokens between sentence pairs are shown in figure 7.3 and 7.4 respectively. The distribution shows that as similarity measure decreases, the number of detected changes increases. This is expected as change between sentence pairs reduces the similarity measure. This also introduces another potential improvement; through further analysis and evaluation on detected tokens and similarity measure, we can fine tune the sentence matching algorithm even further to eliminate more incorrectly matched sentence pairs. High number of detected addition and deleted tokens in the sentence pairs with low similarity measures shows that a threshold can be computed for number of detected changes to eliminate false sentence matches. This can be further extended to identify extreme changes between a policy pair to estimate the applicability of the proposed method for the pair. These extreme changes make comparing the policies using the proposed methods redundant, as the newer version of the policy can be considered a new document.

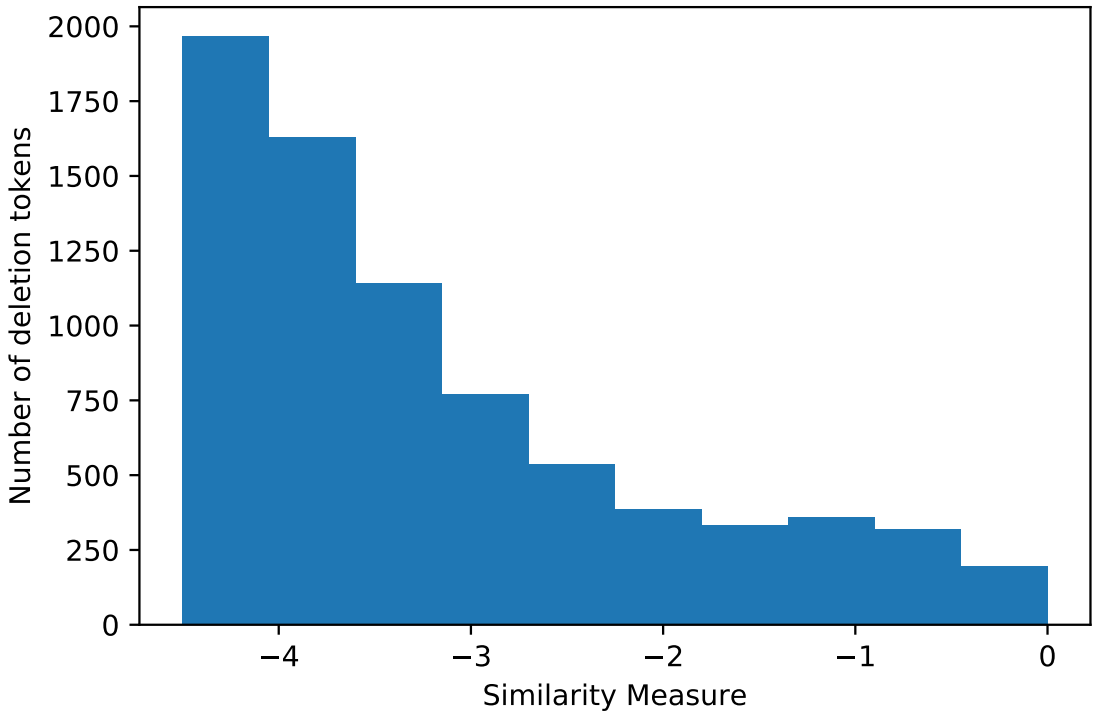

Figure 7.4: Distribution of deletion tokens between sentence pairs

7.5 Web Application

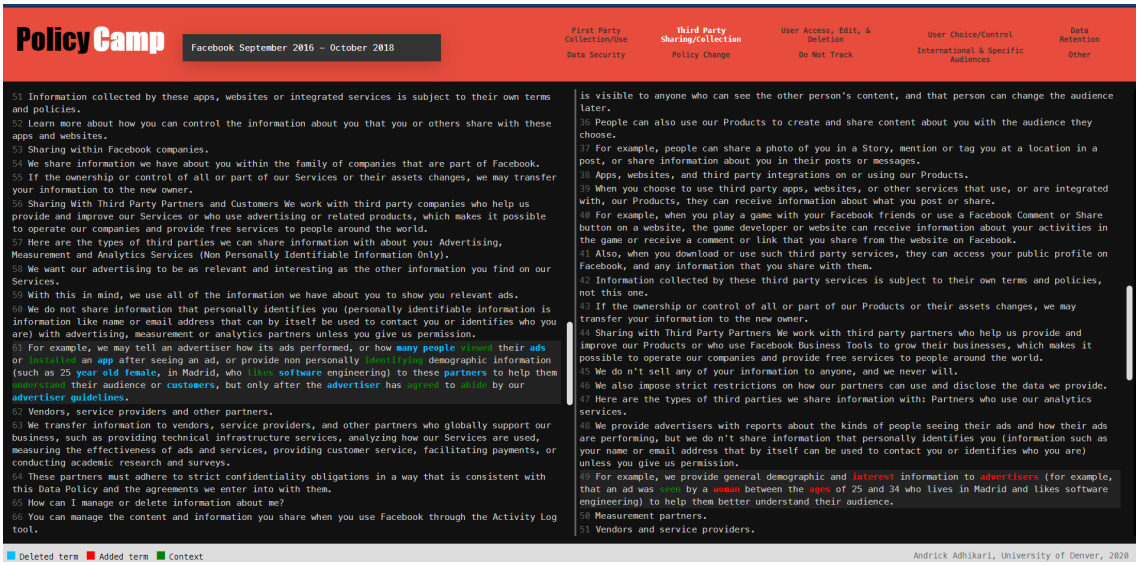

Figure 7.5: A Web application for comparing two versions of a privacy policy

We also implemented a web application that allows user to input two versions of a privacy policy and compare them side by side. Figure 7.5 shows the implemented application. On hovering over a sentence with the mouse pointer, the category of the sentence gets highlighted on the top right corner of the application. The respective match of the sentence in the other version of the policy gets automatically scrolled into view and it's background gets highlighted. The detected changes between the sentences are highlighted with colors. Red shows added tokens, blue shows deleted tokens and green shows the context of the added and deleted tokens. Currently the input files are precomputed using the Python implementation of sentence classification, sentence matching and change detection methods. Files are input using the drop down menu in the top left corner of the application. The application can be improved by implementing proper backend APIs for real time computation of the methods.

8. Conclusion and Future Work

The process of developing tools for comparing two versions of a privacy policy is explored in this research. By using natural language processing and machine learning, we have implemented an application that allows users to compare privacy policies at a sentence level and identify the relevant changes. This can inform users of data practice changes introduced by a company or website, without having to go through ambiguous and incomprehensible policy documents. This was realized by first using a sentence classification method to organize policy texts into expert defined policy practice categories. Sentence classification gave us some interesting insights into privacy policy structures, confirming our concerns raised by the the nature of these documents. We observed that even though policies try to inform users about the companies' data practice, the complicated nature of the documents beats the purpose. Users are given very little control over their own data and the instructions are often not clear. Sentence classification was followed by sentence matching, where we used glove encoding combined with word mover's distance to match the sentence of a policy to the semantically most similar sentence in the next version of the policy. This allows the user to compare the two versions of the policy at a sentence level. We used the sentence categories to not only improve the performance of sentence matching, but also to inform users about the sentences. A threshold was also computed through proper analysis that minimized false sentence matches between two policies. Even though sentence matching is computationally expensive for real time application, this is a one time task only needed when a new policy arrives. There is potential for improvements through modification of the word embedding methods to give more weight to the policy related important terms. A word embedding method tuned specifically for privacy policies can significantly improve the sentence matching precision. Lastly, matched sentences were used as input for our change detection method, which presents the users with addition and deletion changes between the sentence pairs. The method also informs

users about the context of the change. There are many potential future improvements for the change detection method. We only explored the change detection in terms of nouns and the context, which produced informative results. Exploring other part of speech tags and dependency relationships may also present some interesting results in terms of changes. Methods can also be devised that alerts the users of concerning changes in a policy. The changes can also be classified into severity levels that reflects the impact it will have on the user's privacy.

Our developed tool can be improved further. The application should be able to present relevant results to the users without overwhelming them. The tool can be enhanced with a filtering feature that allows the users to view sentences from a particular category. Users should be provided with an option to set alerts for concerning classes of changes in the policy. This could be at a user preferred granularity, say, if the user is interested in addition of newer methods of data collection, or even newer forms of third party data sharing. The tool should be able to generate results according to user preference. Providing users preferences in terms of the 11 privacy principles [14] will be valuable. Through proper back-end implementation, the tool can be given the ability to monitor the privacy policies of user selected websites. Whenever a policy is changed, the tool will automatically generate results for the users and notify them.

Future work on identifying contextual changes may introduce significant enhancement for the tool. Suppose, if the policy changes the sentence "We may share your location information." to "We will share your location information.", a user should be aware of such a change. Change detection method needs to be improved further for identifying relevant contextual change. Not only that, the tool should have the ability to estimate the severity of the change and it's impact on the user's privacy, and efficiently communicate the same to the user.

Nowadays, mobile phones are commonly used for most user needs. That requires our tool to be compatible for mobile phones as well, which is a major task in itself. Migrating the application to mobile phones as a mobile application requires efficient methods that can run in mobile hardware and also present results to the user in a concise format, suitable for mobile phones. Due to limited screen size, a usable user interface for the mobile version of the tool is of utmost importance. The tool should be able to monitor the privacy policies of all the installed applications and notify the user when a change takes place in any of those policies. The same fundamentals can be extended for web browsers as well. An extension for browsers can be implemented that

retrieves and caches the privacy policies of the visited websites. Whenever a particular website is revisited, the privacy policy will be automatically compared to the last cached policy of the website, and notify the users of any change, if present.

An extensive future work will involve working towards a tool that can compare policies across different products of the same company. For example, Google has a cornucopia of products, “Gmail”, “Google Drive”, “Google Lens” etc.; a tool that compare the policies across all the products can provide an even more informed assessment. The work presented here is not only applicable for comparing privacy policies, but can also be extended for other types of natural language documents such as software documentations, legal documents, published work, etc.

Bibliography

[1] Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. “A brief survey of text mining: classification, clustering and extraction Techniques”. In: *CoRR* abs/1707.02919 (2017).

[2] Waleed Ammar, Shomir Wilson, Norman Sadeh, and Noah A Smith. “Automatic categorization of privacy policies: A pilot study”. In: *School of Computer Science, Language Technology Institute, Technical Report CMU-LTI-12-019* (2012).

[3] Fred H Cate. “The limits of notice and choice”. In: *IEEE Security & Privacy* 8.2 (2010), pp. 59–62.

[4] Sudarshan S Chawathe and Hector Garcia-Molina. “Meaningful change detection in structured data”. In: *ACM SIGMOD Record* 26.2 (1997), pp. 26–37.

[5] Sushain K Cherivirala, Florian Schaub, Mads Schaarup Andersen, Shomir Wilson, Norman Sadeh, and Joel R. Reidenberg. “Visualization and interactive exploration of data practices in privacy policies”. In: *Symposium on Usable Privacy and Security*. 2016, pp. 3–10.

[6] Lorrie Faith Cranor. “Necessary but not sufficient: Standardized mechanisms for privacy notice and choice”. In: *J. on Telecomm. & High Tech. L.* 10 (2012), p. 273.

[7] Lorrie Faith Cranor. “P3P: Making privacy policies more useful”. In: *IEEE Security & Privacy* 1.6 (2003), pp. 50–55.

[8] Morgan C Evans, Jaspreet Bhatia, Sudarshan Wadkar, and Travis D Breaux. “An evaluation of constituency-based hyponymy extraction from privacy policies”. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 312–321.

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. “Deep Learning http://www.deeplearningbook.org”. In: *MIT Press, Cambridge, MA* (2016).

[10] Hana Habib, Sarah Pearman, Jiamin Wang, Yixin Zou, Alessandro Acquisti, Lorrie Faith Cranor, Norman Sadeh, and Florian Schaub. “" It’s a scavenger hunt": Usability of Websites’ Opt-Out and Data Deletion Choices”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–12.

[11] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G Shin, and Karl Aberer. “Polisis: Automated analysis and presentation of privacy policies using deep learning”. In: *27th USENIX Security Symposium*. 2018, pp. 531–548.

[12] Mitra Bokaei Hosseini, Sudarshan Wadkar, Travis D Breaux, and Jianwei Niu. “Lexical similarity of information type hypernyms, meronyms and synonyms in privacy policies.” In: *AAAI Fall Symposia*. 2016.

[13] Judith Hurwitz and Daniel Kirsch. “Machine learning for dummies”. In: *IBM Limited Edition* 75 (2018).

[14] ISO/IEC. *Information technology—Security techniques—Privacy framework*. International standard ISO/IEC 29100:2011(E). Geneva, Switzerland: International Organization for Standardization, 2011.

[15] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. “From word embeddings to document distances”. In: *International conference on machine learning*. 2015, pp. 957–966.

[16] Rainer W Lienhart. “Comparison of automatic shot boundary detection algorithms”. In: *Storage and Retrieval for Image and Video Databases VII* 3656 (1998), pp. 290–301.

[17] Seung-Jin Lim and Yiu-Kai Ng. “An automated change-detection algorithm for HTML documents based on semantic hierarchies”. In: *Proceedings 17th International Conference on Data Engineering*. 2001, pp. 303–312.

[18] Frederick Liu, Shomir Wilson, Peter Story, Sebastian Zimmeck, and Norman Sadeh. “Towards automatic classification of privacy policy text”. In: *School of Computer Science Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-ISR-17-118R and CMULTI-17-010* (2018).

[19] Haibin Liu, Tom Christiansen, William A Baumgartner, and Karin Verspoor. “BioLemmatizer: lemmatization tool for morphological processing of biomedical text”. In: *Journal of biomedical semantics* 3.1 (2012), p. 3.

[20] Julie Beth Lovins. “Development of a stemming algorithm”. In: *Mech. Transl. Comput. Linguistics* 11.1-2 (1968), pp. 22–31.

[21] J-F Mas. “Monitoring land-cover changes: a comparison of change detection techniques”. In: *International journal of remote sensing* 20.1 (1999), pp. 139–152.

[22] Michelle McCormick. “New Privacy Legislation”. In: *Beyond Numbers* (2011).

[23] Aleecia M McDonald and Lorrie Faith Cranor. “The cost of reading privacy policies”. In: *Journal of Law and Policy for the Information Society* 4 (2008), p. 543.

[24] Anca Micheti, Jacquelyn Burkell, and Valerie Steeves. “Fixing broken doors: Strategies for drafting privacy policies young people can understand”. In: *Bulletin of Science, Technology & Society* 30.2 (2010), pp. 130–143.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[26] Tom M Mitchell et al. “Machine learning. 1997”. In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[27] Alessandro Oltramari, Dhivya Piraviperumal, Florian Schaub, Shomir Wilson, Sushain Cherivirala, Thomas B Norton, N Cameron Russell, Peter Story, Joel Reidenberg, and Norman Sadeh. “PrivOnto: A semantic framework for the analysis of privacy policies”. In: *Semantic Web* 9.2 (2018), pp. 185–203.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[29] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[30] Rohan Ramanath, Fei Liu, Norman Sadeh, and Noah A Smith. “Unsupervised alignment of privacy policies using hidden markov models”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2014, pp. 605–610.

[31] Joel R Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T Graves, Fei Liu, Aleecia McDonald, Thomas B Norton, and Rohan Ramanath. “Disagreeable privacy policies: Mismatches between meaning and users’ understanding”. In: *Berkeley Tech. LJ* 30 (2015), p. 39.

[32] Norman Sadeh, Alessandro Acquisti, Travis D Breaux, Lorrie Faith Cranor, Aleecia M McDonald, Joel R Reidenberg, Noah A Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. “The usable privacy policy project”. In: *Technical report, Technical Report, CMU-ISR-13-119*. Carnegie Mellon University, 2013.

[33] Kanthashree Mysore Sathyendra, Abhilasha Ravichander, Peter Garth Story, Alan W Black, and Norman Sadeh. “Helping users understand privacy notices with automated query answering functionality: An exploratory study”. In: *Technical report*. 2017.

[34] Kanthashree Mysore Sathyendra, Shomir Wilson, Florian Schaub, Sebastian Zimmeck, and Norman Sadeh. “Identifying the provision of choices in privacy policy text”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017, pp. 2774–2779.

[35] Florian Schaub, Rebecca Balebako, Adam L Durity, and Lorrie Faith Cranor. “A design space for effective privacy notices”. In: *Eleventh Symposium On Usable Privacy and Security*. 2015, pp. 1–17.

[36] Matthew W Vail, Julia B Earp, and Annie I Antón. “An empirical study of consumer perceptions and comprehension of web site privacy policies”. In: *IEEE Transactions on Engineering Management* 55.3 (2008), pp. 442–454.

[37] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N Cameron Russell, et al. “The creation and analysis of a website privacy policy corpus”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1330–1340.

[38] Shomir Wilson, Florian Schaub, Frederick Liu, Kanthashree Mysore Sathyendra, Daniel Smullen, Sebastian Zimmeck, Rohan Ramanath, Peter Story, Fei Liu, Norman Sadeh, et al. “Analyzing privacy policies at scale: From crowdsourcing to automated annotations”. In: *ACM Transactions on the Web (TWEB)* 13.1 (2018), pp. 1–29.

[39] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman M Sadeh, Steven M Bellovin, and Joel R Reidenberg. “Automated Analysis of Privacy Requirements for Mobile Apps.” In: *Network and Distributed System Security*. 2017.

[40] Sebastien Zimmeck. “The information privacy law of web applications and cloud computing”. In: *Santa Clara Computer & High Tech. LJ* 29 (2012), p. 451.