2020

Content-based Filtering Recommendation Approach to Label Irish Legal Judgements

Sandesh Gangadhar
Technological University Dublin

Content based filtering recommendation approach to label Irish legal judgements

Sandesh Gangadhar

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computing (Data Analytics)

5th January 2020

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing(Data Analytics) – DT228A from Technological University Dublin, is entirely my own work and has not been taken from the work of others. Any of the content taken from others work is duly acknowledged and cited within the text of my work.

The dissertation was prepared according to the regulations for postgraduate study of Technological University Dublin and has not been submitted in whole or part for any other Institute or University for an award. The work reported on in this dissertation abides by the requirements and principles of the University's guidelines for ethics in research.

Signed: *Sandesh Gangadhar*

Date: *5th January 2020*

Abstract

Machine learning approaches are applied across several domains to either simplify or automate tasks which directly result in saved time or cost. Text document labelling is one such task that requires immense human knowledge about the domain and efforts to review, understand and label the documents.

The company Stare Decisis summarises legal judgements and labels them as they are made available on Irish public legal source www.courts.ie. This research presents a recommendation-based approach to reduce the time for solicitors at Stare Decisis by reducing many numbers of available labels to pick from to a concentrated few that potentially contains the relevant label for a given judgement. To solve this problem, traditional and state-of-the-art text feature representations along with K-Nearest Neighbour recommender using both cosine similarity and word mover's distance are developed and compared. A series of experiments are designed starting from TF vectors and KNN recommender which is set as a baseline. Further experiments were designed after observing the results of the current experiment. Pre-trained word2vec was used in this experiment as a baseline for state-of-the-art approaches and domain specific embeddings were developed using data scraped from legal text sources.

The results obtained show that traditional feature representations do not suit well for this problem. Pre-trained word embedding with cosine similarity showed greater performance among resultant vector-based recommenders as it was trained on 100 billion words from Google news data. Domain specific embeddings developed were

only trained on 60 million words and henceforth provided the results of around 0.24 precision at max. On the other hand, pre-trained embedding had lower vocabulary intersection of 46% as compared to the domain specific word embeddings which had about 61% at max. Moving forward from cosine-based approaches, WMD proved hard to evaluate as its computational cost is high at this point of time. However, this measure has shown greater potential towards solving this problem by surpassing all of the cosine similarity-based domain specific recommenders and traditional representation-based recommenders on a sample data that was used to evaluate. There was about 5% increase in precision of the recommenders as compared to other recommenders developed including the pre-trained embedding which performed no different with WMD.

The research found out that the product of context by summing the word vectors require high-quality word vectors to capture the analogy of the words when operations are performed with it. This directly correlates with the number of examples provided of each word for the embedding models to learn meaningful word vectors representations. This research highlighted critical aspects of the approach that needs to be addressed to solve the problem such as the importance of similarity/distance metrics for KNN when solving text data, amount of data required to achieve a quality of word vectors, the computation required to solve the dataset used. This paves the way to perform further exploration on the topic.

Keywords: nearest neighbours, word embedding, word2vec, FastText, cosine similarity, word mover's distance, term frequency, tf-idf, recommendation engine

Acknowledgements

I would like to express my sincere gratitude to my dissertation supervisor Sarah Jane Delany, for her patience, knowledge and continuous support throughout this research.

My sincere gratitude to the company **Stare Decisis** for providing me with the labelled Irish legal judgements which were an essential component for this research. I would also like to thank all the professors and fellow students at TU Dublin - city campus for their professionalism and support over the course of my studies.

Contents

List of Figures

List of Tables

List of Acronyms

ML	Machine Learning
KNN	K Nearest Neighbour
NN	Neural Network
NLP	Natural Language Processing
WMD	Word Mover's Distance
CBoW	Continuous Bag of Words
TF	Term Frequency
TF-IDF	Term Frequency Inverse Document Frequency
RecSys	Recommendation System
EMD	Earth Mover's Distance
CBF	Content Based Filtering
UBCF	User Based Collaborative Filtering
IBCF	Item Based Collaborative Filtering
LOOCV	Leave One Out Cross-Validation
NDCG	Normalized Discounted Cumulative Gain

Chapter 1

Introduction

1.1 Background

The invention and utilization of computing devices are constantly simplifying tasks that generally took significantly more time when performed by humans. The advancement in technology and the ease of access to digital platforms has resulted in a massive explosion in the amount of text data created and stored. These texts can be in many forms across several domains such as web articles, emails, legal documents, medical documents, Facebook posts, tweets, resumes, business documents *etc.* The human society is completely dependent on information and this only keeps increasing going forward. Lots of information lying around and more coming every second is great but is only useful if we can make sense of them and utilize it to grow as a society. Irrespective of the purpose and domain in a broader sense labelling these unstructured text data can help achieve better organization, retrieval and searches which can increase how one can use them meaningfully. However, the process of labelling the documents is tedious and requires a significant amount of human effort. Just as with many other problems that were solved, computers can be used in an attempt to solve this problem. The field of machine learning (ML) that comes under the broader field of computer science has constantly been used for decades to solve similar kind of problems. ML also has affiliations with other fields such as mathematics and statistics.

"A field of study that gives computer the ability to learn without being explicitly programmed" (Samuel, 1959).

The above statement defining ML was stated in the year of 1959 by Arthur Samuel who is credited with creating self-learning computer programs at IBM. The field of ML has matured significantly since then. In ML several types of algorithms and their ever-growing variants are designed and improved constantly to solve variety of problems. In general, ML algorithms attempt to find a function for a problem that it is trying to solve while making sure it is as close to the realistic version of the function as possible. For example, human beings naturally have the ability to read the text written on pretty much anything and understand the writings. The process with which human solve this is that by receiving light from the surface of where the writings are present through the eyes and our brain will process this light to form an image and extract a meaning out of it which is text. Here, anything that goes between the reception of light (input variable) to understanding the text (output variable) is the function or in general an ability. ML algorithms attempt to model such abilities by learning from different types of input and output pairs. Once modelled these functions or models can be used to predict the output variable using new input whose output is not known yet.

Back to labelling, when a human evaluator assigns labels for the documents it is of a possibility that he is likely to use certain labels more often than others and some of the labels not assigned of the bunch could still be relevant for the context of the documents. This introduces subjectivity or bias in the labels assigned. There has been a significant advancement in the techniques developed through research and more research is still being conducted to make the process easy and as objective as possible. The labels assigned is very similar to how the words are used in the language where small number of words are used more often than many others during communication(Cancho & Solé, 2003).

The input variables that are provided to the ML algorithm to model the problem are also called features. Features are a representation of the raw data that they were derived from and several types of representations may be available for the same data. Since computers are designed to work with numbers and not characters all of the text data must be converted to numbers before performing any type of operation. Hence, researchers develop ML models with different types of representation in an attempt to solve the problem. There are several types of feature representations available for text data and each come with their own advantages and disadvantages.

The traditional ones are the Bag of Words(BoW) based feature representations such as term frequency(tf) and term frequency-inverse document frequency(tf-idf). They are both memory and computationally intensive and capture only the syntactic meaning while completely ignoring the semantic meaning of the text.

Word embeddings is a collective term for a class of ML models that transform words or phrases to real-valued numerical vectors called as word vectors. The current state-of-the-art in text feature representations is based on word embeddings. They address the issues that existed in traditional representations such as sparsity, capturing semantic meaning and reduction of dimensionality. Domain specific text will be key while developing word embeddings as word vectors obtained will align more towards legal literature.

The field of ML has several branches that are used to solve different type of problems. Classification is one of the branches of ML and another branch recommendation system is also closely related. Classification approaches are generally rigid and goal-oriented and tend towards finding the exact labels. Recommendation approaches, on the other hand, are more relaxed versions of the classification approach. Instead of searching for exact labels they provide a bunch of recommendations in decreasing order of importance in the hope to contain the relevant labels. Since the legal documents used for this research has labels varying from 1 to 9 per document and considering the subjectivity that may be introduced while labelling the documents recommendation

approach seems to be a suitable fit to solve this problem.

K Nearest Neighbours(KNN) is one of the simplest to implement and easy to understand ML algorithm. It simply stores all of the labelled cases and uses a similarity measure between the features of unlabelled and labelled cases to recommend labels for unlabelled cases. There are several similarity metrics available and cosine similarity is the popular choice. However, the state-of-the-art for document similarity at the moment is Word Movers Distance (WMD) by (Kusner, Sun, Kolkin, & Weinberger, 2015) which operates on word vectors from word embeddings. The labelled dataset covering about 6900 judgements is provided by Stare Decisis for this research. Stare Decisis is an online service company that provides summaries for all the new judgements from the Irish superior courts. They cover every new judgement from the Irish supreme court, Court of appeal and the high court. Their services include easy searches by keyword, topic, court or judge for the Irish legal documents. They manually label each new judgement document that is published. The source judgements are available on a public website `www.courts.ie` without the labels and as of 20th October 2019 there are about 13000+ judgements.

1.2 Research Problem

Legal documents or in general any text documents are written in natural language in a human-understandable format which varies with the linguistic expressions of the authors. Legal text, in general, will have its own vocabulary and style of writing which is very different from general text data. Labelling these kind of documents must be carried out appropriately using Natural Language Processing (NLP) based techniques while focusing on the domain. Solving this for Stare Decisis using a recommendation system approach is a challenge as development and evaluation of both traditional and state-of-the-art feature representation and similarity measures will require a number of individual models to be built and compared. Different feature representations and similarity metrics along with a selection of K for KNN recommender will again offer

its challenges.

1.3 Research Aims and Objectives

This research aims to develop an ML-based solution to reduce the amount of manual effort required to label the Irish legal documents. In technical terms, this research aims at finding a suitable text feature representation and similarity measure for KNN based recommender that can be utilized to solve the problem of labelling the Irish legal documents. This will reduce significantly the amount of manual effort required to review the documents and understand the available labels to use them to label the unlabelled document by the solicitors at Stare Decisis.

The objectives of this research are as follows:

1. Acquire labelled dataset from Stare Decisis.
2. Perform data cleaning and make it ready for feature extraction.
3. Extract tf and tf-idf vectors for traditional and word vectors from word embeddings for state-of-the-art feature representation from the legal documents.
4. For embeddings, two sets of word vectors will be extracted; one from Google's pre-trained word2vec embedding and new embeddings created using data scraped from several legal sources.
5. Implementation of KNN recommender using both traditional and state-of-the-art similarity measures and feature representations. Cosine is the traditional similarity measure while WMD is currently the state-of-the-art.
6. Performing a series of comparisons on the developed recommenders to identify the suitable similarity measure and feature representation to label Irish legal documents.

1.4 Research Methodology

This research is empirical in nature and is designed to gain knowledge through experimentation and testing the feasibility of the approaches in consideration. The research is secondary in nature as it will utilize existing research in the domain of NLP and ML to evaluate content-based filtering recommendation system to label Irish legal judgements using the labelled dataset procured from Stare Decisis.

Lastly, the research is inductive in nature as experiments are conducted in an attempt to solve the research problem.

1.5 Research Limitations

Bias associated with the labels assigned could impact the evaluation of the recommender. The recommender developed and evaluated in this research will still be on the grounds of the initial distribution of labels.
The computational complexity of WMD is (p^3 $logp$) (Le & Mikolov, 2014) where p represents the number of words in the corpus. Applying this is still a challenge to perform this with computational power available today when comparing two long documents. Hence, experiments are designed to evaluate only on a small sample of the dataset.

This research limits itself to default settings with word embedding algorithms unless mentioned due to computational complexities of the algorithms and time available. Open-source tools are used to develop this research and other enterprise-scale tools such as Amazon EC2, Google cloud, Microsoft Azure ML are not considered.

1.6 Document Outline

The remainder of this document is broken into the following chapters:

1. Literature Review gives an overview of data mining, frameworks, approaches

to feature engineering as well as approaches to automatically build and tune machine learning pipelines. This will focus the work around the identified gap.

2. Experiment Design & Methodology will detail the experiments proposed to address the research questions. It will define the methods, libraries and tools that will be used to test the hypothesis as well as the metrics required to evaluate the experimental results and ultimately answer the research questions.

3. Implementation & Results will detail all the work done in order to complete the experiments. It will also present the experimental results of the various modelling approaches.

4. Evaluation and Analysis will interpret and analyse the results obtained during the experiments, finally evaluating the proposed research question.

5. The conclusion chapter will summarise the totality of the work undertaken and the results obtained. It will also identify any future work that could be considered.

Chapter 2

Literature Review

This chapter will discuss about the all the components of the research. It will start with the concept of machine learning itself and drill down to the finer details of the components that are relevant to this research. It will also discuss the aspects surrounding the concepts related while providing an understanding of the field, the trends associated with the components, approach and the methodologies as found in the current literature.

2.1 Machine learning

There are several well stated definitions for machine learning (ML), the below is the popular one stated by Tom Mitchell in his book titled "Machine Learning"(Mitchell, 1997).

"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."

For example, a computer program that learns to play board games such as "Chinese Go" may improve its performance as measured by its ability to find patterns and win games at a class of tasks involving playing the game, either through experience obtained by playing against humans or itself. In general, to have a problem that is

well defined one must identify these three segments: the class of tasks, to improve the measure of performance, and the source of experience.

Following Mitchell's definition, the basic pipeline for ML as shown in 2.1 is to take a collection of input and output variable pairs called as training data to develop a model, which must then be objectively evaluated before making inference on unseen data.

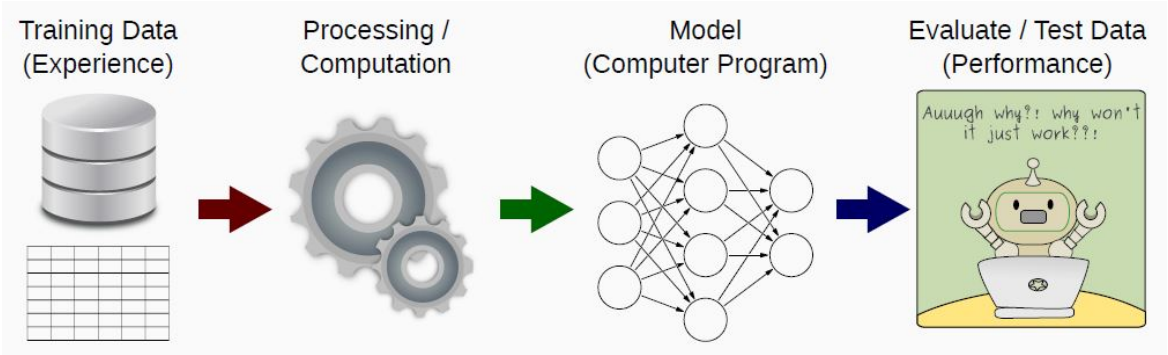

Figure 2.1: Machine learning pipeline

(Source: Machine learning SPEC9270 at TU Dublin-City campus by David Leonard)

In the real world, most of the data require some sort of pre-processing before it can be used along with ML algorithms. Pre-processing generally include multiple steps of operations on the data such as extraction of features, the transformation of features(mathematical operations, scaling etc.), handling missing data and many more and varies wildly from data to data. Algorithms are supplied with suitable pre-processed data called training data to develop the model. Once developed, another set of data called validation data is used to evaluate its performance and suitability for the purpose.

2.2 Types of Machine Learning

The field of machine learning is broad and contains several branches. The two most popular branches are supervised and unsupervised learning. There is a slight intersection between the branches and this area is called as semi-supervised learning.

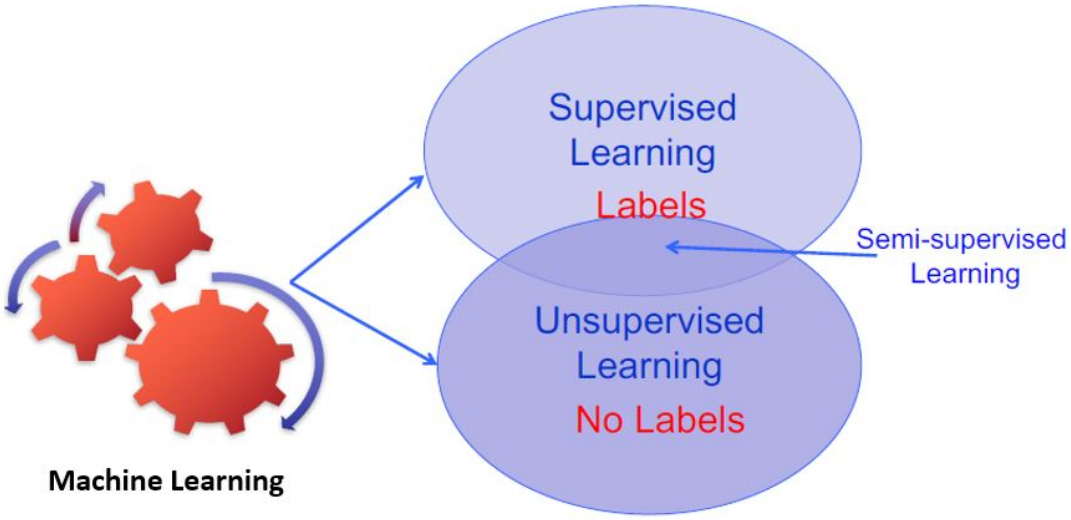

Figure 2.2: Supervised and unsupervised learning
(Source: Machine learning SPEC9270 at TU Dublin-City campus Sarah Jane Delany)

There are other techniques such as recommendation systems which overlap across several domains such as ML, information retrieval *etc*. For this reason, when research authors speak about types of ML, supervised and unsupervised are generally chosen and branches such as recommendation systems are not often discussed. It is often confusing with recommendation systems to assign a type as they can be either supervised or unsupervised or mix of both in terms of ML language.

2.2.1 Supervised learning

Supervised learning is a process of developing a model using input-output pairs to learn patterns and use this to make inference on unseen input instances. The input-output pairs used to learn patterns are called as a training set and the process of identifying patterns in them is called learning. The process of model performance evaluation generally starts by keeping some sets of input-output pairs away from training set called test set. Once the learning phase is complete the model developed is used to make predictions on the test set. The predictions are then compared with the actual output and evaluation measures are calculated according to the type of task.

The supervised learning further branches out depending on the type of tasks. The key distinction being the type of outcome variable to be predicted. There are two types of supervised learning, classification and regression.

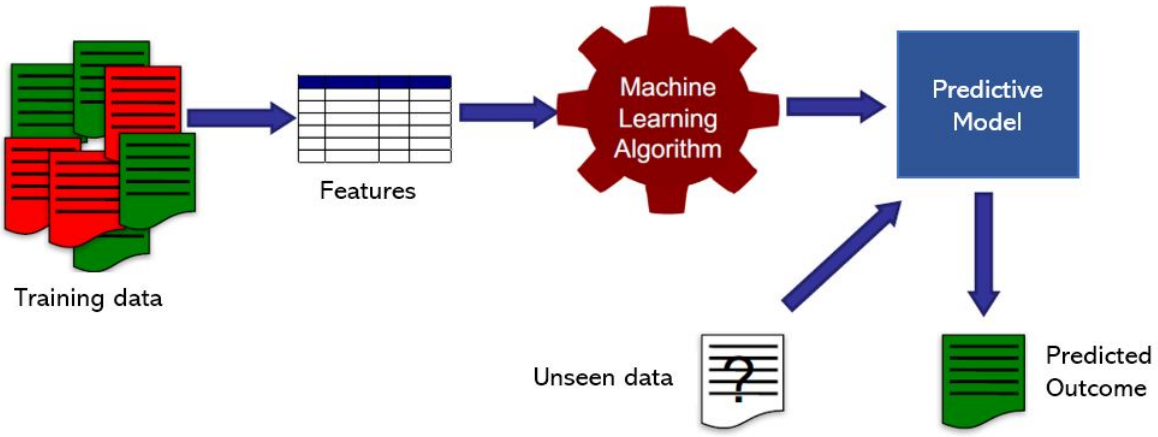

Figure 2.3: Supervised learning
(Source: Machine learning SPEC9270 at TU Dublin-City campus Sarah Jane Delany)

The above figure 2.3 shows the general workflow of supervised learning which remains the same for both classification and regression. However, the type of outcome variable will distinguish one from the other.

Classification

The outcome variable for this type of task is discrete in nature such as ("yes" or "no"), ("High", "Medium", "Low") and so on. There are three major types of classification. *Binary classification* is a type of classification task where the outcome variable has two distinct classes and any given input can belong to any one of the class. Example: ("yes" or "no"), ("high" or "low"), ("male" or "female") *etc*

Multi-class classification is where the outcome variable has more than two distinct class and any given input can belong to anyone of these classes.
Example:("High", "Medium", "Low"), ("Large", "Medium", "Small") *etc.*

Multi-level classification is a type of task where a given input can belong to more than one of the available classes.

Example:

• Image: labels - (table, computer, window and sky),

• Web article: tags - (philosophy, fiction, science) *etc.*

Some of the most popular classification algorithms are, support vector machine(Yang, 2019d), multi-layer perceptron(Witten, Frank, Hall, & Pal, 2017), K Nearest Neighbours (Yang, 2019b), Decision trees(Duda, Hart, & Stork, 2012), Logistic regression (Yang, 2019a) *etc.*

Regression

The outcome variable for this type of problem is continuous in nature such as (age of a person), (income), (time duration) *etc.*

Some of the popular regression algorithms are, linear regression (Ross, 2017), multi-layer perceptron, decision trees, K Nearest Neighbours, support vector machine *etc.* Some of the algorithms mentioned above can be used to perform both classification and regression tasks.

2.2.2 Unsupervised learning

With unsupervised learning, there is no defined outcome variable and the task of the algorithm is to discover any pattern from the data that is interesting. Clustering is an example of an unsupervised learning technique that is used to find clusters within data such that the data points inside the cluster are homogeneous and the datapoints outside the cluster are heterogeneous to each other ideally.

Some example of clustering algorithms are k-means (Witten et al., 2017), hierarchical clustering (Theodoridis & Koutroumbas, 2009), DBSCAN (Kotu & Deshpande, 2019b) *etc.*

Dimensionality reduction is another example of unsupervised learning technique. It is

a process of reducing the number of available variables, by obtaining a set of principle variables that explain majority of the variance. Some of the popular dimensionality reduction algorithms are principal component analysis (PCA) (Yang, 2019a), linear discriminant analysis (LDA) (Yang, 2019a) *etc.*

2.2.3 Recommendation system

Recommendation system (RecSys) refers to any system that is capable of recommending a set of items for a given input hoping to contain the relevant items.
More formally, RecSys can be defined as a decision making strategy for situations where the target for a particular input can belong to multiple classes with conflicting opinions among the users (Rashid et al., 2002). The concept of RecSys is broad and spreads across several domains such as ML, information retrieval, and *etc.* The goals of RecSys are nearly similar to multi-label classification. Multi-label classification aims at predicting exact labels for a given input while RecSys on, the other hand, takes a more relaxed approach with predictions in the form of multiple recommendations.

Figure 2.4: Recommendation example

(Source: Created using `https://www.storyboardthat.com/storyboard-creator`)

As shown in the figure 2.4, in some cases input can belong to multiple available labels and level of agreement between the association of labels to input can vary from user to user. RecSys will be a great fit in such environments which works by reducing

the range of options available to few related options by eliminating the ones that are irrelevant.

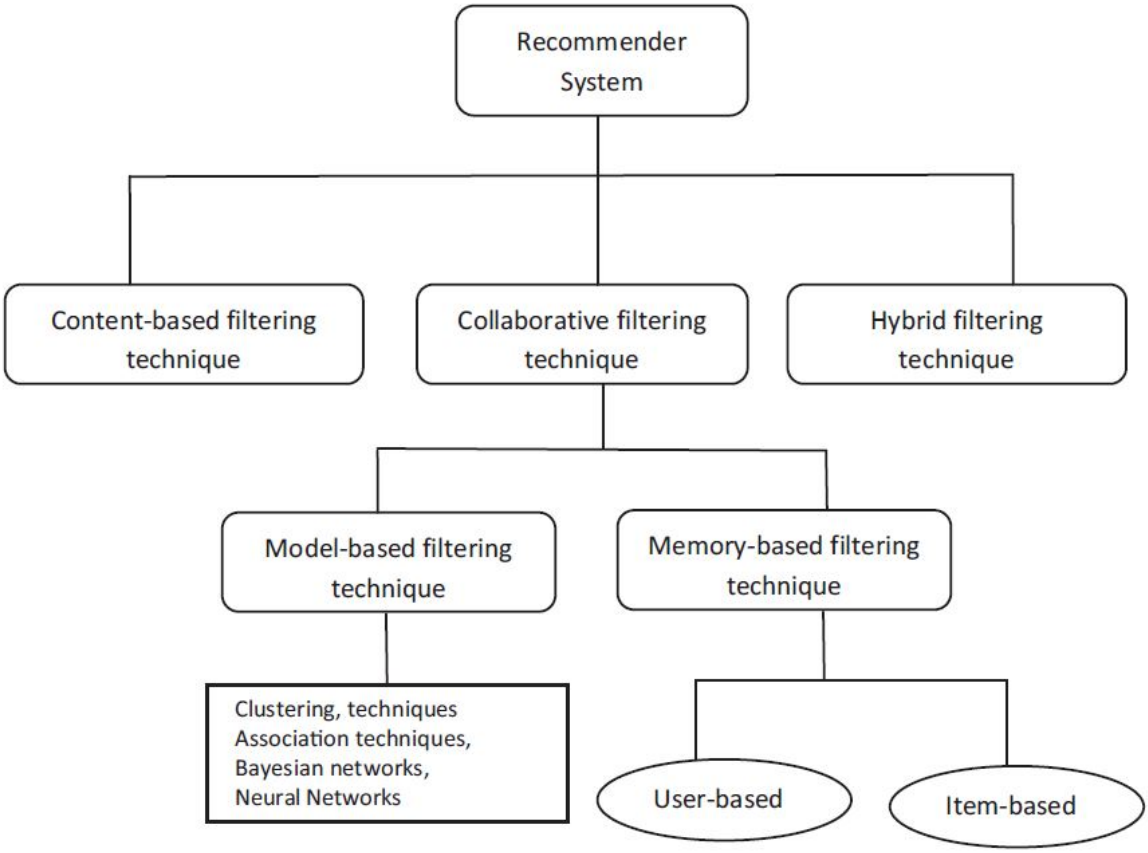

Figure 2.5: Classification of recommendation systems
(Source: http://www.sciencedirect.com/science/article/pii/S1110866515000341)

RecSys can be classified into three major categories, content-based filtering, Collaborative filtering and Hybrid filtering as shown in the figure 2.5.

Content based filtering(CBF)

It is a domain dependent technique where emphasis is more on the analysis of the attributes of the items in order to make recommendations. (Isinkaye, Folajimi, & Ojokoh, 2015) suggest that when text documents such as publications, web articles

and news are to be recommended CBF is most appropriate and successful. Here, recommendations are made using features extracted from content of the items that user has interacted with, in the past. In order to make meaningful recommendations CBF makes use of similarity between documents using different approaches. To model the difference between the documents CBF can use vector space models such as tf, tf-idf (Kotu & Deshpande, 2015), word embedding vectors or probabilistic models such as Naïve Bayes classifier (Friedman, Geiger, & Goldszmidt, 1997) or Decision trees. Either ML techniques or statistical analysis can be used be make recommendations by learning the underlying model.

Collaborative filtering(CF)

It is a domain independent technique used for content such as music and movies where they cannot be adequately or easily described using metadata. It works by creating a database such as user-item matrix to hold preferences of users. CF can be further classified into memory-based and model-based approaches (Gronvall et al., 2018; Bobadilla, Ortega, Hernando, & Gutiérrez, 2013).

Memory based techniques have found widespread success due to their effectiveness in real life applications. These can be further divided into user based and item based collaborative filtering. In user based collaborative filtering(UBCF) for any given *userA* his nearest neighbours are calculated based on their profile and ranked and top nearest neighbours are selected. The items that are rated by these users except for the ones that *userA* has already rated are ranked and recommended to *userA* (Schafer, Frankowski, Herlocker, & Sen, 2007).

Item based collaborative filtering(IBCF) is the opposite of UBCF where it calculates similarity between items to identify potential users.

Model based techniques learn a model in an order to improve the performance of the CF technique using previous interests of users. This approach can eliminate the sparsity problem that memory-based approach using dimensionality reduction techniques. Learning algorithms are used to analyse the user-item matrix to identify the relationship between items. Model learnt can be employed to make recommendations

for new users.

Hybrid filtering

Different techniques can be combined to gain better recommendation performance or overcome the limitations and problems of pure recommendation approaches.

This research utilizes the similarity-based approach on text representations of legal documents to recommend labels. The recommender technique used in this research is Content Based Filtering. The impact of feature representation type and the similarity techniques are explored in this research in an attempt to find a suitable approach to generate relevant recommendations.

2.3 Text feature representation

Text feature representation is of one the key components of this research. This section will discuss the different types of text representation that are used in this research and also the popular approaches that are found in the literature as of date.
There are several types of feature representations available for text data and each of them come with their own advantages and disadvantages. The traditional ones are the Bag of Words(BoW) based feature representations such as term frequency(tf) and term frequency - inverse document frequency(tf-idf) while the state-of-art approaches are mostly word vector based.

2.3.1 Bag of words

In this method, firstly a vector with the number of dimensions equal to the number of distinct words from a set of documents called corpus is created. Each element of the vector representing a particular word. This vector is extracted from each document

where the vector elements represent how many times the corresponding word has appeared in that document.

Example:

Doc A = "This is an apple. Apple is ...";

Doc B = "This is an orange. Orange ..."

Doc/ Words	this	is	an	apple	orange
BoW(*Doc A*)	1	2	1	2	0
BoW(*Doc B*)	1	1	1	0	2

Table 2.1: Example: Bag of Words

2.3.2 Term Frequency - Inverse Document Frequency

Term frequency and inverse document frequency is also known as tf-idf. It is a product of two statistics related to the text. The weights represented by tf-idf represent how important a word is to a document in a corpus(collection of documents).

Term frequency(TF)

It measures how frequently a term has occurred in a document and this alone can be a good representation for the text documents.

$tf = \frac{N_D}{T_D}$

$N_D = Number\ of\ times\ word\ w\ occurs\ in\ a\ document$

$T_D = Total\ number\ of\ words\ in\ the\ document$

Suppose if the length of *Doc A* is 100 words and *"the"* appears 15 times out of 100 then the word *"the"* holds a weight of 0.15.

Inverse Document Frequency(IDF)

IDF defines the importance of the word based on its frequency in the corpus. The idea is to reduce the weights of the words that appear frequently across the corpus. Meaning the words that appear most frequently such as "the", "is", "that", "it" and

so on will provide little to no meaning during text analytics.

$idf = \log_e(\frac{T_C}{D_w})$

$T_C = Number\ of\ documents\ in\ corpus$

$D_w = Number\ of\ documents\ with\ word\ w\ in\ it$

Suppose if there are 100 documents and the word *"the"* appears 200 times in total then the *idf* of the word will be -0.69.

Term frequency inverse document frequency(tf-idf)

Finally, tf-idf is an amalgamation of the both tf and idf. Multiplying both the tf and idf will provide with tf-idf weights for each words in the document. For the word "the" as per the examples given above it will be,

tf-idf(Doc) = 0.15 X (-0.69) = -0.1035

As observed the weight of the frequent word "the" decreased from 0.15 to -0.1035.

2.3.3 n-Grams

An n-gram is a sequence of n characters or words (Aiyar Shetty, 2018). First, the value of n is chosen and then the combination of n characters or words in sequence as appeared in corpus is extracted. The representations such as BoW, tf, tf-idf can also be calculated for these combination.

Example:

Doc A: "this is an apple"

Assuming the words in the corpus as the same as in *Doc A*. The below is the representation of BoW of *n-grams* where n=2.

Doc/n-Grams	this is	is an	an apple
BoW(*Doc A*)	1	1	1

Table 2.2: Example: n_Gram

2.3.4 Word vectors

Measuring a semantic relationship between text documents plays a vital role in a variety of language processing tasks such as plagiarism detection, finding similar documents and so on. Semantic similarity is challenging due to varying linguistic expressions of the authors.

Vector space models have been used in distributional semantics for decades now. Since then number of models have appeared for estimating continuous representation of words such as latent dirichlet allocation (Bastani, Namavari, & Shaffer, 2019) and latent semantic analysis(Yu, Xu, & Li, 2008) to name a few.

Word embeddings are an application of neural networks (Fathi & Shoja, 2018) that takes words from vocabulary of corpus as input and translates them to lower dimensional space and applies back-propagation (Eberhart & Shi, 2007) techniques to fine-tune the weights. Word embeddings are the weights of first layer of the network, which is usually referred to as embedding layer or projection layer. The embedding layer of a neural network can project all of the words in the corpus to just a few nodes. The weights of these projection layer capture the semantic meaningful representation of the words. There were several pseudo implementations of this concept (Bengio, Ducharme, Vincent, & Janvin, 2003) but none that were feasible on corpus with large vocabulary due to lack of computational power. First successful implementation on large vocabulary was demonstrated by (Mikolov, Chen, Corrado, & Dean, 2013) with millions of words.

Word embedding is a collective term for models that learn to map words in the vocabulary to numerical vectors. This technique reduces the dimensions of the text data while providing additional benefits such as capturing semantic meaningful representation of the words. There are 3 popular vector space representation under word embedding, namely, word2vec, FastText and GloVe.

Word2vec

(Mikolov, Chen, et al., 2013) introduced a neural network based distributed representation of words. This is arguably the most popular of the available word embedding

models. It is a two-layer shallow neural network that is designed to process text data. The input provided to this is a text corpus while output received is a set of feature vectors for words in that corpus. The authors have recommended two architectures to calculate word vectors namely, Continuous Bag of Words(CBoW) and Skip-gram.

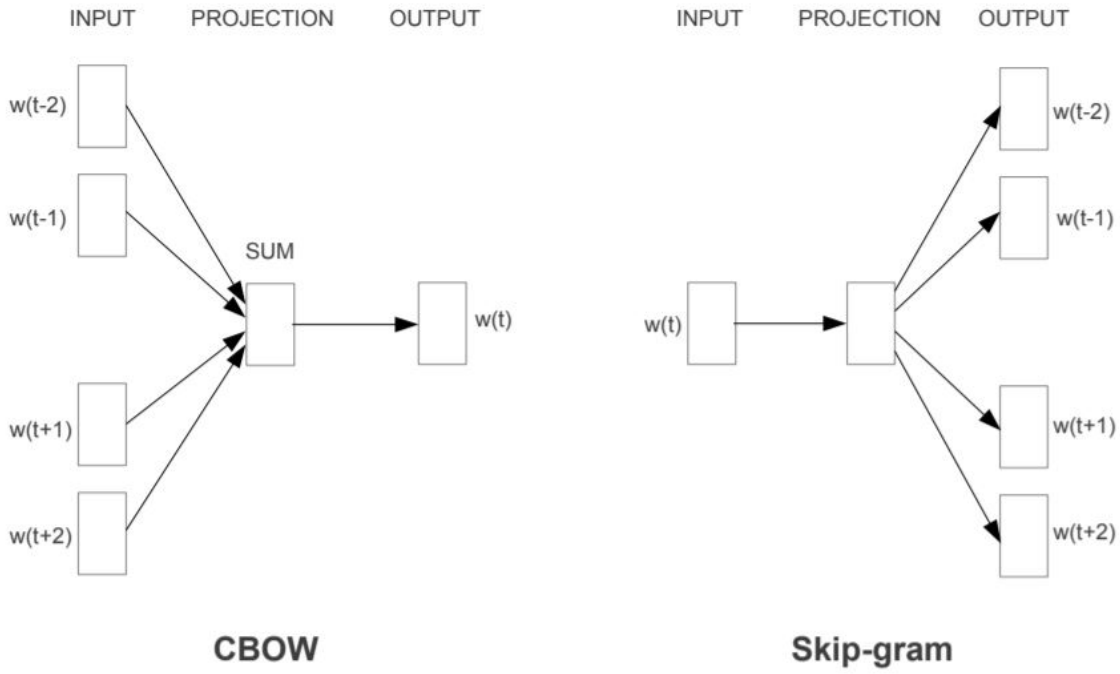

Figure 2.6: CBoW and Skip-gram architectures
(Source: (Mikolov, Chen, et al., 2013))

CBoW architecture is designed to predict the current word based on the context. The architecture used here is a classic feed-forward neural network. The neighbouring words with a chosen window size are provided as an input to predict the current word. Excluding the hidden layers, the linear projection layer is shared and their index corresponding to each word will be a numeric word vector for that word.

All of the words in the corpus will be converted to a set of binary vectors whose length is equal to the length of the vocabulary. The position of the value 1 in the vector indicates a particular word in the vocabulary. This method of representation is also called as on-hot encoding. The one hot encoded vectors of neighbouring words are used as input while predicting the target word which is also a one-hot vector. The

model is trained to predict the target word by sliding the window across the text in the corpus several times while adjusting the weights. Numerical representation of words can be extracted using the weights of the projection layer which broadcasts all of the words in the vocabulary to a defined set of nodes generally in the number of few tens to hundreds.

Skip-gram is another approach to achieve word embeddings. Here projection layer will be present at the output layer as compared to CBoW method whose projection layer was at the input. Skip-gram aims to train a feed forward neural network to predict the neighbouring words given a target word.

Words are represented by real valued vectors, typically in the range of few tens to hundreds of dimensions, as opposed to dimension equal to the size of the vocabulary in traditional BoW-based representations such as TF/TF-IDF (Aizawa, 2003). The distributed vector representation of the words is learned depending on the usage of the words as found in the corpus. The more the examples of the usage in different sentences the better the representation of the vectors. This allows similar words to have similar representations naturally capturing the meaning of them. This is how the problem of synonyms encountered in traditional methods is solved.

The application of word2vec extends beyond text processing, it is also used in recommendation systems, gene encoding, social media graphs *etc.* The obtained embedding also allows the user to perform vector calculus like addition and subtraction to find different meanings and relationship of the words.

The figure 2.7 provides a general idea behind how vector addition is performed in vector space. When two vectors are added a new vector is formed and this is called as a resultant vector. In the example, the resultant force is used to show the direction in which the swimmer is pushed considering his direction of swim and the direction of the current.

Similarly, embedding assigns direction for each word in vector space where similar words face similar direction and opposite words face the opposite direction. When vector calculus is performed on the word vectors meaning analogies can be expected. One of the famous demonstrated example of word embedding is when you perform

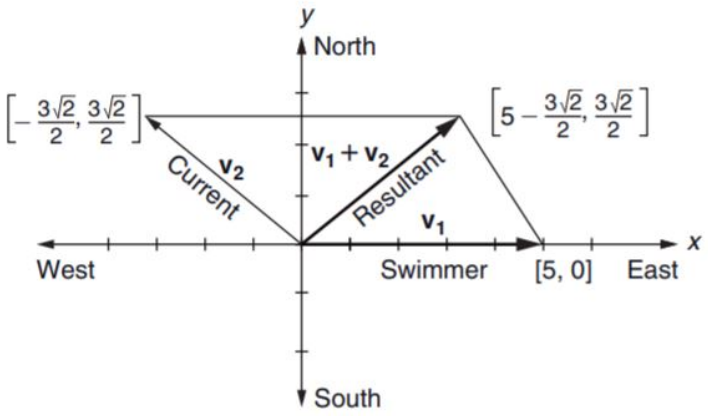

Figure 2.7: Resultant vector

(Source:

`http://www.sciencedirect.com/science/article/pii/B9780123747518000019`)

the arithmetic operation of word vectors (King – Man + Woman) you get Queen on relevant word embedding. This example demonstrates analogical reasoning capability of word embeddings. (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) have clearly explained that addition of two vectors results in product of their context and the product here works similar to an "AND" operation. They have also demonstrated that large amount of quality data to create embeddings results in better performance. There are several algorithms to generate word embeddings and some of the popular are word2vec, GloVe and FastText.

FastText

FastText is another implementation of word2vec but with some architectural changes (Bhardwaj, Di, & Wei, 2018). Unlike word2vec FastText also considers the internal structure of the words while learning word representations. The smallest unit to train on for word2vec is words themselves while on the other hand for FastText it is character level n-grams. For example, for the word "happy" the word vector with n-gram of minimum size 3 and maximum size 6 can be decomposed to:
lt;ha, lt;hap, lt;happ, lt;happy, hap, happ, happy, happygt;, app,appy, appygt;, ppy, ppygt;, pygt;

Due to this FastText generates better word embeddings for words that are infrequent. Even the infrequent words will have many neighbours here as supposed to word2vec where they are left stranded away with little to no neighbours. When faced with new words which are not found it the vocabulary word2vec completely halts and since FastText is on the character level as long as the n-grams of these words have appeared in the training corpus FastText will be able to construct word vectors by summing up character n-gram vectors. The disadvantage of using FastText to extract word vectors is, as this algorithm works on the character level it takes large memory and time to learn the embedding. Since this is an extended version of word2vec both CBoW and skip-gram variants are available. Pre-trained FastText embeddings are available at `https://FastText.cc/` in 157 different languages.

2.3.5 Other representations

There are several other feature representations available for text and also many that are derived from the ones explained already. Some of these representations are explained below.

PCA on BoW representations: As explained earlier, BoW based representations suffer from sparsity and dimensionality issues. Hence, dimensionality reduction techniques are applied on these to reduce both problems. Principle Component Analysis (PCA) is one of the popular dimensionality reduction technique. Given a data d with variables l, PCA determines a subspace of dimension $m \leqslant l$, such that after projection of the variables into this subspace the statistical variance of the variables is optimally retained. The subspace is defined by m mutually orthogonal axes called as principle components. These principle components capture the majority of the variance from l in decreasing order. Hence, just choosing a small number of principle components from the start will drastically reduce the number of dimensions while explaining majority of the variance from the initial data d.

Doc2Vec: It is an extension of word2vec models whose aim is to create a numeric representation of document, irrespective of its word length (Le & Mikolov, 2014). The word vectors represent the concept of word the document vectors represent concept of document. The slight addition was made to the already existing word2vec with a feature vector which is document unique. During the training of word2vec document vector is also trained and holds the numeric representation of the document at the end of the training.

Selecting characteristic words: (B. Li & Han, 2013) have an interesting idea of utilizing the characteristic(feature) words selected from the document calculated by tf-idf. This establishes a relationship between the labels which are presented as legal provisions and features which are as mentioned the characteristic words. The method to attain characteristic words is through the application of the Chi squared statistic during the traditional tf-idf. Then the position of the characteristic words is calculated in the document by introducing correction factors through CHI squared test and integrate them with the tf-idf values. They have proven that by introducing this additional process into the flow provides a significant improvement in the model performance over the traditional method. The author explains that this improved version of the tf-idf method solves the problem of distribution of the feature words between the target classes and insufficient importance of the keywords when tf-idf alone is applied.

2.4 Algorithms

An algorithm is a finite set of well defined, computer instructions that takes some value, or set of values as input and produces some value, or set of values, as an output. It is thus a sequence of computational steps that transforms input to output (Cormen, Leiserson, Rivest, & Stein, 2009).

To solve the problem of labelling Irish legal judgements using recommender approach there are several available algorithms that are relevant. Depending on the type of

feature representation used appropriate approaches can be utilized to solve the task. The key ideas behind some of the relevant algorithms for this research are discussed in detail below.

2.4.1 K Nearest Neighbour (KNN)

K-Nearest Neighbours(KNN) (Yahyaoui's, Yahyaoui, & Yumuşak, 2018a) is a type of supervised learning algorithm. It is one of the easiest to implement and understand. This algorithm stores all of the provided input and uses similarity measure to classify new input.

Given a set of training data T, a distance measure D and an integer K. For a new data point p whose output is unknown, the algorithm searches T for K nearest points using measure D and assigns the common output among its neighbour to p.

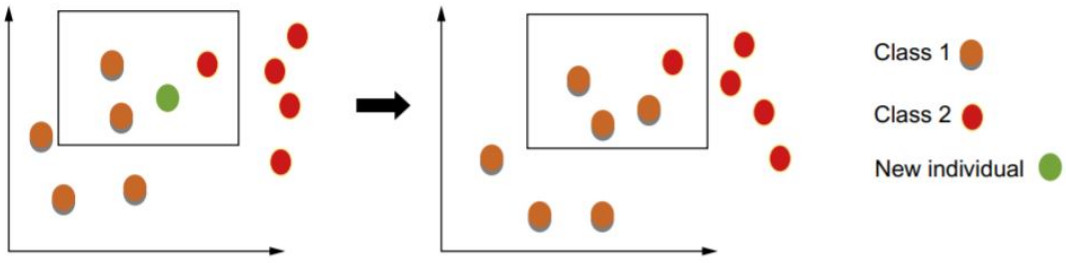

Figure 2.8: Principle of K Nearest Neighbour algorithm

(Source:

`http://www.sciencedirect.com/science/article/pii/B9780128131855000097`)

As shown in the figure 2.8, for the classification of a new data point the distance D is measured between it and all the other data points in T. In the provided in the image above 3 nearest neighbours are chosen for the new data point and Class 1 is provided as the prediction.

KNN can also be used to perform regression instead of selecting the most popular outcome as the outcome for new data point as in classification an average or median can be taken and provided as the prediction to perform regression tasks. The Content Based Filtering recommendation engine is basically a KNN based approach.

The performance of KNN hinges on the determination of similarity and dissimilarities between the memorized data point and the new data point. To quantify this range of measures are available such as calculating distance, correlation, cosine similarity and Jaccard similarity (Kotu & Deshpande, 2019a).

Rescaling the variables is one of the key pre-processing steps that have to performed when dealing with similarity measures. If the variables present in a data point is of different scales such as kilogram, milligram and microgram the measure will be right on individual variable level and completely wrong when averaged. For example, if a difference between two data points at different scale was (Variable1: 0.01kilogram, Variable2: 1 gram,Variable3: 1000 milligram). Then the average taken of the differences can easily be dictated by any of the particular variable on smaller units. Here in this case the average considering its numbers as it is stands at 333.67 which is completely false. If each of the values were normalized to one scale say, grams then the true value stands at 4 grams. Hence, normalization also called as scaling is a crucial step to be performed when working with algorithms that use either distance or similarity metrics.

Distance measures

The popular distance measures used are Euclidean, Manhattan and Hamming.

Euclidean:

If X and Y are two data points with k features, X = (x1, x2, ... ,xk-1, xk) and Y = (y1, y2, ... ,yk-1, yk) then their distance can be measured along the straight line between the two in k dimensional space using Pythogorean theorem. (Kotu & Deshpande, 2019a)

$$Euclidean(\text{X,Y}) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan:

It is the sum of the difference between the variables of X and Y instead of root of squared distance. This distance measure is also called as taxicab distance, as this is similar to the visual path traversed by a vehicle around city blocks. (Kotu &

Deshpande, 2019a)

$$Manhattan(\mathrm{X,Y}) = \sqrt{\sum_{i=1}^{k} |(x_i - y_i)|}$$

Hamming:

Euclidean and Manhattan distances work best with numerical variable but hamming distance suits well with categorical variables especially binary variables. It measures the number of components by which two vectors or strings differ (Yang, 2019c).

Word mover's distance:

Word mover's distance (WMD) is a completely different approach to solving the text similarity problem. Instead of using two vectors to calculate the similarity between documents which has been done so far with TF, TF-IDF and resultant word vectors along with cosine, WMD uses word vectors for each word from both the documents. At the core of WMD is another metric called as earth mover's distance(EMD) which is used to solve transportation problem. EMD is a measure of distance between two probability distributions over a region. Informally, consider two separate heaps of sand distributed differently over the ground and EMD calculates the minimum amount of work done to transport one heap to look like the other. WMD similarly, transports words from one document to the other in vectors space created by embedding. WMD calculates the minimum amount of work to transport words from document one to document two.

Word embeddings capture a meaningful representation of each word and this is the key to calculate WMD. Words similar in meaning will be closer and transportation of such words require less work, in other words will have smaller distance. Dissimilar words will have more distance and transportation cost is high. To identify similar words between the documents WMD has to calculate distance of travel for each word from document one to document two and vice versa. Hence, the computational cost to calculate WMD is extremely high and with the best average time to solve this problem scales to $O(p^3 \log p)$, where p denotes unique number of words in the documents.

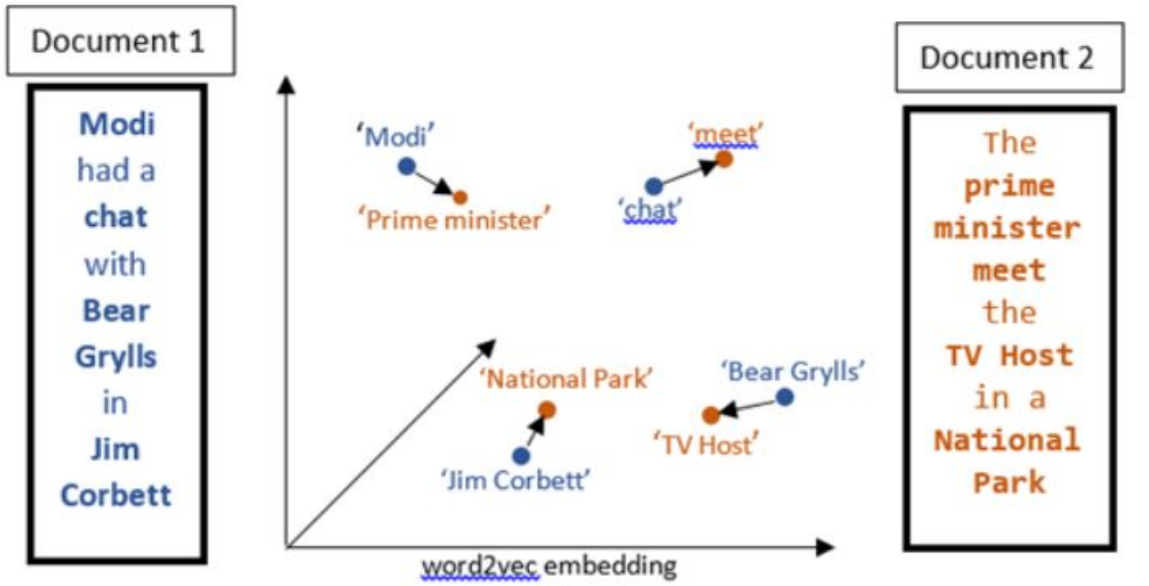

Figure 2.9: Word mover's distance

(Source: `https://towardsdatascience.com/word-movers-distance-for-text-similarity-7492aeca71b0`)

Similarity

Unlike distance measures, similarity measures are generally range bound and few of the popular similarity measures are Jaccard and cosine.

Jaccard similarity:

Jaccard similarity also known as Jaccard Index. For a given sets X and Y Jaccard similarity is the measure of the ratio between the number of items in both the sets over total number of distinct items from both the sets (Metcalf & Casey, 2016).

$$Jaccard(\mathrm{X,Y}) = \frac{X \cap Y}{X \cup Y}$$

Cosine similarity:

Given two non-zero vectors X and Y cosine similarity measures as the name suggests the cosine of the angle between them in an inner product space (Metcalf & Casey, 2016).

$$Cosine(\mathrm{X,Y}) = \frac{X.Y}{||X||||Y||}$$

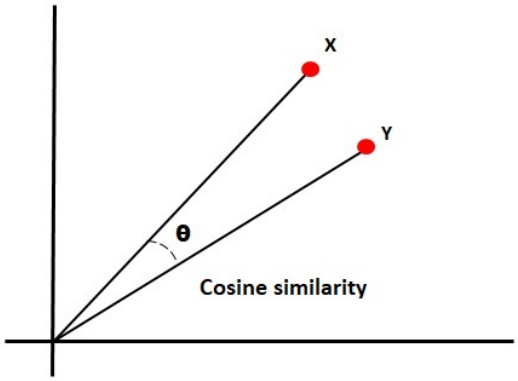

Figure 2.10: Cosine similarity

(Source: https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/eb9cd609-e44a-40a2-9c3a-f16fc4f5289a.xhtml)

As the name suggests, it measures the cosine of the angle between two non-zero vectors. The value of cosine similarity ranges from -1 to 1. Two vectors with same direction will produce the value of 1 indication they are similar and two vectors in opposite direction will produce the value of -1 indicating they are dissimilar. Cosine can be used with BoW representations or the word vector representations to identify similarity between the text documents.

2.4.2 Decision tree

Decision trees are one of the supervised learning algorithm. It is a classification approach-based data induction learning (Shi, 2014). There are several version of this algorithm available and the popular ones are ID3 and C4.5.

Terminology

Root node: point of access or first node.

Branch: A link between two nodes

Leaf: a terminal node

Internal node: a node that is not a leaf.

Construction and working

The construction and design of decision tree below is provided as per the version ID3.

Decision trees as the name suggests builds a classification or a regression model in the form of an inverted tree structure. It breaks down given training data D with n variables to k smaller sets di using a particular test criterion as calculated using a statistical measure of some sort in an attempt to separate the observations present in D to sets called as a pure set. A pure set is a subset of the any given initial data which contains only the observations for one type of outcome class. Child node is created for each split made on test criterion and the observations in parent node is shared among child nodes. The level of purity keeps increasing from parent node to child node. Again, calculation of criterion will be done on child node and the process continues recursively on each child node created to attain pure nodes. It is possible to create more than one decision tree for the same training data. The identification of criterion to split the data is the key aspect for the construction of the decision tree. To identify the purity of the split different measures have been identified such as Entropy, Gini, Classification error, variance and so on. Pruning is a method that is used on decision trees to avoid overfitting on the training data. If-then rules can be extracted from the final decision tree algorithm if required and any predictions made is easy explainable and the rules themselves will self-explanatory.

The adaptation of decision tree algorithm is also used to perform multi-label classification (Vens, Struyf, Schietgat, Džeroski, & Blockeel, 2008) for any given validation data.

2.4.3 Random Forest

An ensemble is defined as a group of complementary parts working together to contribute to a single effect. Random forest is a type of ensemble learning, here multiple decision trees are constructed which often improve the performance over an individual

classifier. There are two popular approaches to create an ensemble called bagging and boosting.

Bootstrap aggregation also known as bagging, here instead of having single training set D of size n. As shown in figure 8, multiple training sets of size n is created by sampling the same data with repetition. In context to random forest multiple decision tree models are developed on these training sets and the predictions are made using the voting mechanism. Recommendations can be made using the labels that were available after voting mechanism.

Boosting is an alternative approach which also uses voting mechanism to make predictions. However, during voting, unlike bagging which weights all of the models equally boosting weights models according to performance. Boosting is an iterative and adaptive method, here models are created and evaluated and next the succeeding model will be encouraged to become and expert for observations which current model has misclassified. The intuition being that all of the models must be made experts to complement each other. Recommendation can be generated using this approach during voting mechanism.

2.5 Evaluation

2.5.1 Metrics

Different evaluation metrics are available for each type of machine learning tasks and the same metrics may or may not be used for the same task as the goals of the research will vary. Regression and classification are types of supervised learning and constitutes a majority of applications of machine learning. Metrics such as recall, and precision are easy to implement and understand and are useful in multiple classification and recommendation tasks. This section will focus on metrics associated with classification which are also used for recommendations.

Classification metrics

The concept of confusion matrix is used to derive metrics associated with classification. The confusion matrix can be used for binary, multi-class and multi-label classifications and also to evaluate the performance of recommender systems.
Evaluation of binary classification provides a general idea of confusion matrix and the same is adapted suitably for other discrete outcome prediction models.

		Actual Class (Observation)	
		Y	**N**
Predicted class (expectation)	Y	*TP* correct result	*FP* unexpected result
	N	*FN* missing result	*TN* correct absence of result

TP, *true positive;* FP, *false positive;* FN, *false negative;* TN, *true negative.*

Figure 2.11: Confusion matrix
(Source: : Data Science(Second edition) concepts and practice by Kotu Deshpande, 2019)

The structure of a binary classification confusion matrix is shown in the figure 9. It is a form of truth table that is traditionally arranged in the form of 2 X 2 matrix. The predicted classes are horizontally arranged in rows and actual classes vertically in columns, the order may be reversed sometimes. Quick way to understand this table is to scan along the diagonal starting on the top left. In an ideal case there will be number only along this axis and zeros elsewhere. The cells on the diagonal are the correct predictions of the model which others are incorrect.
Elements of confusion matrix:
True positive (TP): These are the cases that were predicted to be true and were true in actual.
True negative(TN): There are the cases that were predicted to be false and were false in actual.
False positive (FP): These are the cases that were predicted to be true but were false

in actual.

False negative (FN): These are the cases that were predicted to be false but were true in actual.

These four measures are used to create some of the commonly used metrics to explain the performance of the model. Some of the popular metrics are precision, recall, accuracy, F1 score.

Precision: It is defined as the proportion of the outcomes predicted that were relevant.

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

Recall: It is defined as the proportion of the relevant cases that were found among all the relevant cases.

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

Accuracy: Ability of the classifier to classify all the correct cases as correct and incorrect cases as incorrect.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

F1 score: F1 score is the weighted harmonic mean of precision and recall. Its value tends to be smaller of the two values unlike arithmetic mean. To achieve a high F1 score both precision and recall should be high. This measure is used for the comparison of the models as it incorporated both precision and recall. It's value ranges from 0 to 1. F1 is considered as the appropriate measure when there is imbalance in the label distribution.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The above measures can be used for either binary or multi-class classification tasks.

The adaptations of these measures and others are used for multi-label classification and recommendation engines. Unlike classification, in some recommendation tasks the order of the recommendations will be of utmost important and even these must be taken into consideration when calculating the evaluation metrics.
Precision@N: When dealing with recommendation systems there are multiple outcomes that will be of interest. In this case precision values for each recommendations can be averaged to get an estimate of the model's precision.
Recall@N: Similar to precision@N, but here recall is averaged to get a single number estimate of model's recall.

NDCG: Normalized discounted cumulative gain (NDCG), unlike precision and recall it also considers the order of recommendations made. In cases like search engine top recommendations will be of more value than the ones in the bottom. Hence, correct prediction of the relevant outcomes and their rankings will be of high priority. There are two measures calculated prior to reaching NDCG, Cumulative gain(CG) and discounted cumulative gain(DCG). CG sums up the relevance of the top ranked items while DCG discounts for items that are down the order. NDCG as the name implies is the normalized version of DCG such that range always stays between 0.0 and 1.0. NDCG is an important measure in cases where order of the recommendations is of priority such as search engines, recommendation of the products to customers and so on.

2.5.2 Methods

During development of machine learning model, one needs to evaluate the model in multiple places. The first phase involves with the prototyping where multiple models are tried to identify the best one. Once a suitable model is identified it will be deployed in production where further testing will be made on the live data.

Online evaluation measures live metrics of the model in production on live data.
Offline evaluation measures metrics of the model during prototype stage(and some-

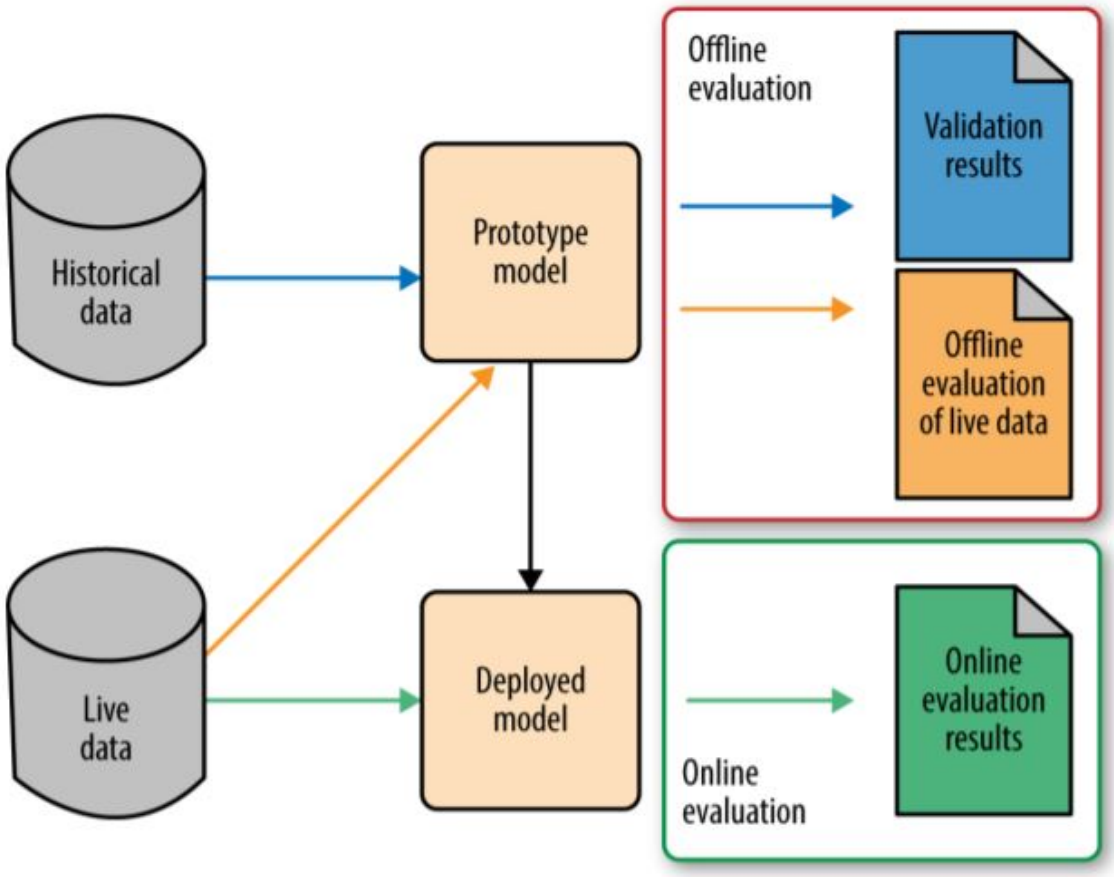

Figure 2.12: Evaluation methods

(Source: : Evaluating machine learning models by Zheng, 2015)

times same metrics on live data as well).

Once trained ML models must be evaluated on a dataset that is independent of the training set. Because the performance measurement on the training data is extremely optimistic estimate of its true performance on new data. A fair evaluation of the model can only be obtained on the data that the model has not seen yet. This approach provides a generalized estimate of the model performance, saying how well the model generalizes to new unseen data.

There are three popular methods of offline model evaluation namely, holdout, cross-validation and bootstrapping.

Holdout method

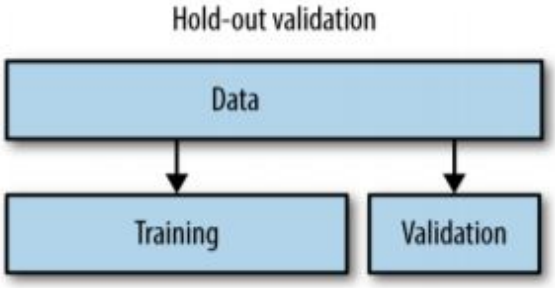

Figure 2.13: Holdout validation

(Source: : Evaluating machine learning models by Zheng, 2015)

It is one of the simplest ways of evaluating the model. Here, assuming all of the data points are independent and identically distributed, for a given data D randomly some data points of D is sampled and held out as validation set. Models are then trained on the training set and evaluated on the validation set. This method is fast and easy to implement but with the downside that it is less reliable. As the evaluation results are obtained from a small subset of data D it is difficult to estimate the generalized metrics, variance information or the confidence intervals. Hold out methods are generally applied when there is a large number of data points available and a subset can be held out and this subset is again large enough to derive statistical estimates of the metrics.

Cross validation

Cross-validation is another validation method and is simply the way of generating different sets of training and validation sets for the process of hyper-parameter tuning. There are many variants of cross validation and the most popular one is k-fold cross validation. In this method the given data D is divided into k subsets. Here, *k-1* subsets are merged to form a training set while one subset is held out as validation set and this process repeats for each subset. The required metrics are calculated during all the k iterations. The overall performance metrics are measured using the average of metrics captured during each iteration.

Another variant of cross validation is leave one out cross validation. This version is

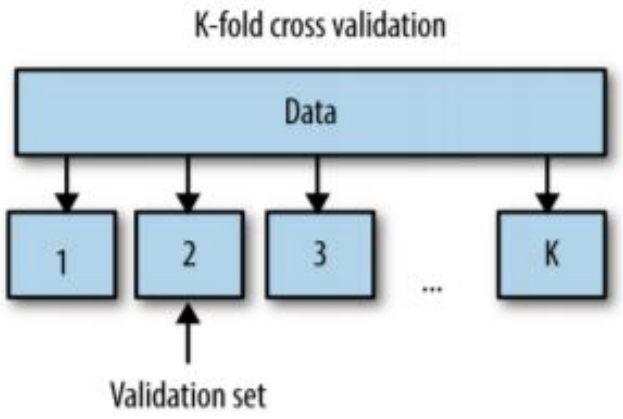

Figure 2.14: k-fold cross validation

(Source: : Evaluating machine learning models by Zheng, 2015)

not but the k-fold cross validation taken to the extreme. The value of k here is equal to the number of data points in the dataset. Meaning, each data point is held out once and the rest of the data points are used as training to develop the model and evaluation metrics is calculated on the held-out data point.

Cross-validation is useful technique to apply when given data D has small number of data points and hold out process is not affordable. It is computationally expensive for large datasets.

Bootstrap aggregation

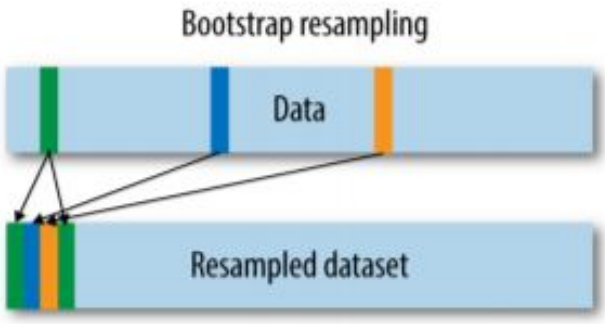

Figure 2.15: Bootstrap validation

(Source: : Evaluating machine learning models by Zheng, 2015)

Bootstrap is a sampling technique. Given a dataset D_A with n observations, it creates

another dataset DB with n observations by uniformly sampling at random D_A with replacement. Since, the real distribution of the datasets is generally not known and only one dataset is provided to represent the population which provides with an empirical distribution. The bootstrap re-samples from the same dataset because sampling without replacement changes the empirical distribution after each draw. Thus, bootstrap sampled set is expected to contain same data points multiple time and the expected ratio of distinct points is (1-1/e) ≈63.2%. In other words, about 63.2% of the data constitute as training set while others that are not selected considered as validation set.

2.6 Types of problems in legal domain

Most of the related and surrounding technical concepts for this research are explained so far in this chapter. The type of problems that are faced in the legal domain are several and with such voluminous text data created efficient management can help speed up several cases in the legal domain. The below are some of the tasks that are performed on legal text.

2.6.1 Summarization

Legal documents are generally lengthy and very different than regular text documents and may contain sentence structure and vocabulary that is not common outside of this domain. It is essential for lawyers and citizens to do an exhaustive research on the related case before they can understand and answer questions in the court. For quite some time legal editors are hired to create judgement summaries to pick out useful information from the chosen judgements to create summaries. Creating summaries of number of documents is tedious and requires significant human effort. Number of ML techniques are developed to solve this problem. The main challenge of summarizing the judgements is about creating short summaries without changing the main context of the lengthy document. The field of text summarization is vast and achieving a text

summary could be done is many ways.

2.6.2 Predictive coding

With the increase in volume of evidences and the cost of manual review, many legal teams consider computer assisted review to help scan through digital records quicker and with less human intervention. Predictive coding is the common name for this procedure which is emerging as an accepted practice in some extremely large cases.
It is a process of using keywords, filtering and sampling to automate segments of e-discovery document review. The goal of predictive coding is to reduce the number of irrelevant and non-responsive documents that need to be reviewed manually. Software tools developed using mathematical model or machine learning model are used to scan and locate data that is relevant to legal cases. Users can use a set of documents to identify potentially relevant documents by training the computer program to identify similar ones. To guide the process and measure effectiveness of the procedure it will generally incorporate statistical or sampling techniques alongside human review. Predictive coding is a type of technology assisted review (TAR) and both of them are used interchangeably. Predictive coding may involve several type of tasks to be performed such as clustering, search models, classification, *etc.*

2.6.3 Document labelling

Lawyers or any user of legal domain could greatly benefit from an organized set of documents either to use them or cite them in their current cases. Recommendation or classification techniques can be used to label legal judgements which is one of the activities that help organize the documents. The task of document labelling also comes under technology assisted review (TAR).

Chapter 3

Design and methodology

This chapter provides the blueprint for this research and acts as the glue between the components explained in the previous chapter. The chapter is structured as follows. The first section of the chapter is the dataset section, it deals with the complete nuts and bolts of the dataset that is received from **Stare Decisis** and the dataset that is scraped to develop word embeddings. Later, section 3.2 will discuss the design of the research. Here explanation about how multiple experiments are performed in an attempt to solve the research problem is given. Lastly, section three is the section dedicated to the methodology of the research which explains about the systematic way of connecting components of the research starting from data to recommendations and everything in between.

3.1 Dataset

Two datasets are used in this research. Dataset 1, the labelled legal dataset was provided by Stare Decisis. Dataset 2, on the other hand, is the one that is scraped as part of this research to develop word embeddings.

3.1.1 Labelled legal judgements

Stare Decisis provided this dataset for this research in a *JSON* file format. When parsed into a table this contained 15 columns with 6981 observations. The column names are presented below,

1. Abstract	6. content	11. key_quotations
2. case_citations	7. judgement_summary	12. Link
3. case_name	8. list_cat	13. primary_cat
4. citation_number	9. primary_judge	14. title
5. primary_court	10. date	15. id

Table 3.1: Labelled legal dataset columns

Of all the columns that are available and as highlighted in the table above only content and list_cat columns are necessary for this research. *Content*, as the name suggests, contains the complete text report of judgement and the list_cat contains the labels to which the report is assigned. Since many labels can be assigned to any given judgement the column list_cat contains multiple labels in the form of single string separated by a comma. The text provided in the Content column is in an HTML format and must be parsed to receive the actual judgement data. Before that, investigation of labels showed that there was a label called *"uncategorized"* that is present independently for 90 of the judgements and 17 cases where it was found alongside other cases. The decision was made to remove these 90 judgements as these cannot be used to evaluate the recommenders developed and in other 17 cases, just the label *"uncategorized"* was removed from the set of labels assigned for judgements.

Contents

There is a certain number of pre-processing steps that have to be applied to the text present in the *Contents* column. The steps taken vary slightly for both the traditional and the state-of-the-art feature representations. The difference being that stemming, and removal of stop words and rare words are only applied to the traditional repres-

entation, apart from this every other step taken is the same.

• Parsing the HTML text to extract the judgement text

• Remove all escape characters (such as , ,̊), punctuation, single character words, extra white spaces.

• Convert the whole document to lowercase text.

• Stemming using porter stemming algorithm. (For only traditional feature representations)

• Removal of stop words and rare words. (For only traditional feature representations)

• Tokenize sentence to words.

The judgements of the legal case in this research can also be referred to as a report or more technically a document. However, not be confused with the usage as they can be used interchangeably. Firstly, exploration of the text data in the context of words and later on the documents.

The figure 3.1 shows the distribution of word frequencies. To generate an informative chart word frequency scale is transformed to log10 scale and to make explanation easy word ranks are converted to percentile. Word rank is just an index assigned to each word, here the min is 1 and the maximum is the total number of distinct words in the corpus which is 56027 and percentile shrinks this to range between 0 to 100. As observed, the frequency distribution of the words are highly skewed with about 31% of the words having a frequency of just 1 for the whole corpus and are considered rare words. On the other end of the spectrum are the most common words with frequency as high as 314784. Going by the Zipf's law (Powers, 1998) the words from the both the ends of the spectrum can be clipped off as rare words are too rare and most common words are too common to add any value. Some words from both the ends of the spectrum given below just to confirm before clipping.

Some rare words from the corpus: Antoin, antifungal, anther, antenatally, answerability *etc.*

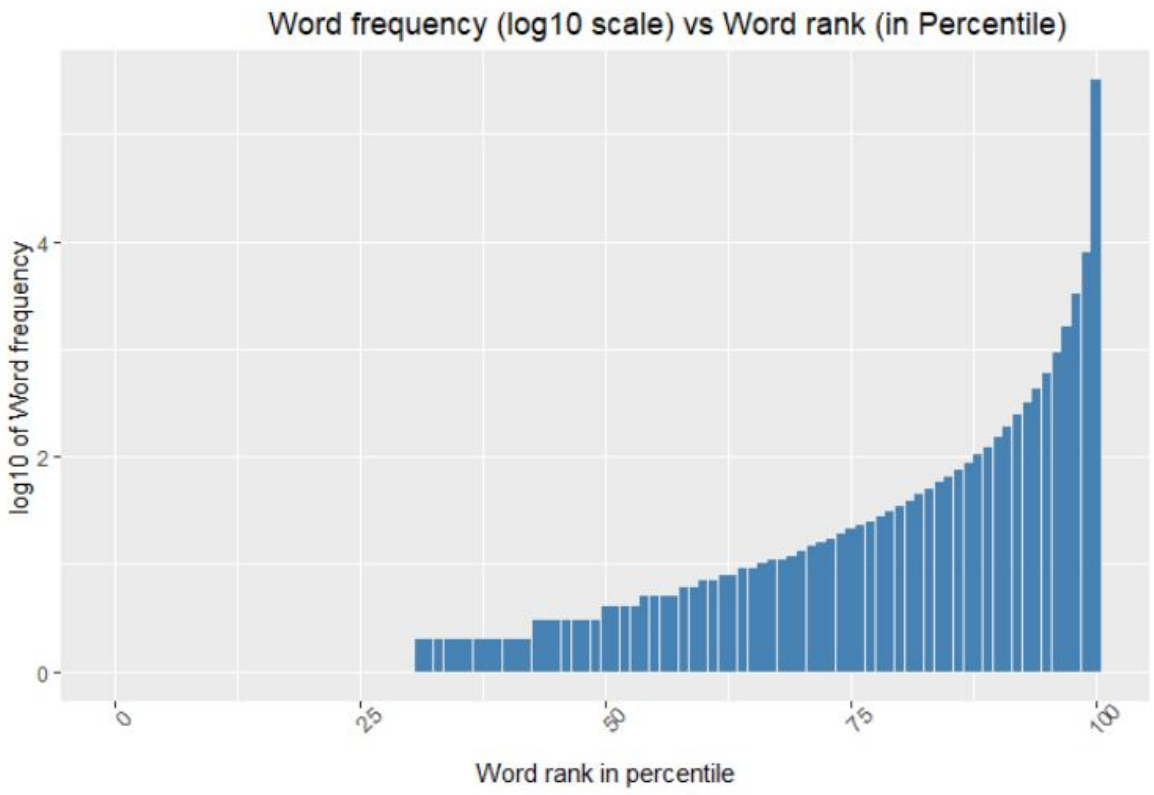

Figure 3.1: Word frequency vs their rank

Some common words from the corpus: And, that, in, to, of, the, review, year and *etc.* The left end of the spectrum is mostly the mix of nouns and rare words while the right end of the spectrum is mostly stop words and words common in legal vocabulary. After observation, the words below 31st percentile and words above 99th percentile will be clipped from the documents. Something interesting happens with the documents after the clipping of the both the left and right ends of the spectrum together removes about 17000 words from the corpus vocabulary. This reduces the count to around 36900 words which completely nullified 2471 documents. There is a critical decision to be made here to either remove the words and together the nullified documents or retain and proceed. The major issue is due to too many infrequent words. To look deeper at the issue, for BoW based feature representations and cosine similarity-based approach removes these words even if included during the dot product and has no effect during the recommendation for other documents. But, the real problem of inclusion is during the evaluation where recommendations of these null documents can offset the

real evaluation scores. To keep the research fair, nullified documents will be removed, and evaluation will be performed. This problem does not exist with the word vectors, irrespective of similarity measure. It is possible that some of the infrequent words could be synonyms with each other if so, will cluster together in embedding space and FastText has greater advantage in this situation as this operates on character level. Now that each documents are pre-processed completely; the details on the documents level can be analyzed. The total number of reports after removal of "uncategorized" is 6891 and later removal of null documents is 4420.

As observed in the table 3.2 the distinct words of either 56027 before removal or 36905 after removal suggests the length of the traditional feature vector for each document will be long. With the minimum of either 6 or 3 and maximum of either 37437 or 17578 and an average of either 2904 or 1358 words per document offers the glimpse of the sparsity of these vectors. This is where a word vector representations shines, as they provide lesser dense vectors for each document while capturing the semantic meaning which traditional feature representations completely ignore.

Measure	Before removal	After removal
Minimum words per document	6	3
Minimum distinct words per document	6	3
Maximum words per document	37437	17578
Maximum distinct words per document	2530	2530
Average words per document	2904	1358
Average distinct words per document	695	489
Median words per document	2206	994
Median distinct words per document	643	432
Total words in the corpus	≈20 Million	≈ 6 Million
Total distinct words in the corpus	56027	36905

Table 3.2: Statistics of text documents before and after removal of words

Labels

Labels are the outcome variable of each document and this research is performed to recommend labels to new documents. The data provided contains 130 different types of labels of which one is *uncategorized* as mentioned earlier and was removed from the dataset. The number of labels from here on is 129.

Figure 3.3 is the frequency chart of all the labels assigned for judgements. This chart clearly shows that some of the labels are very frequent while most of the others are infrequent. This could possibly be an indication of the presence of bias as discussed earlier in chapter 1.

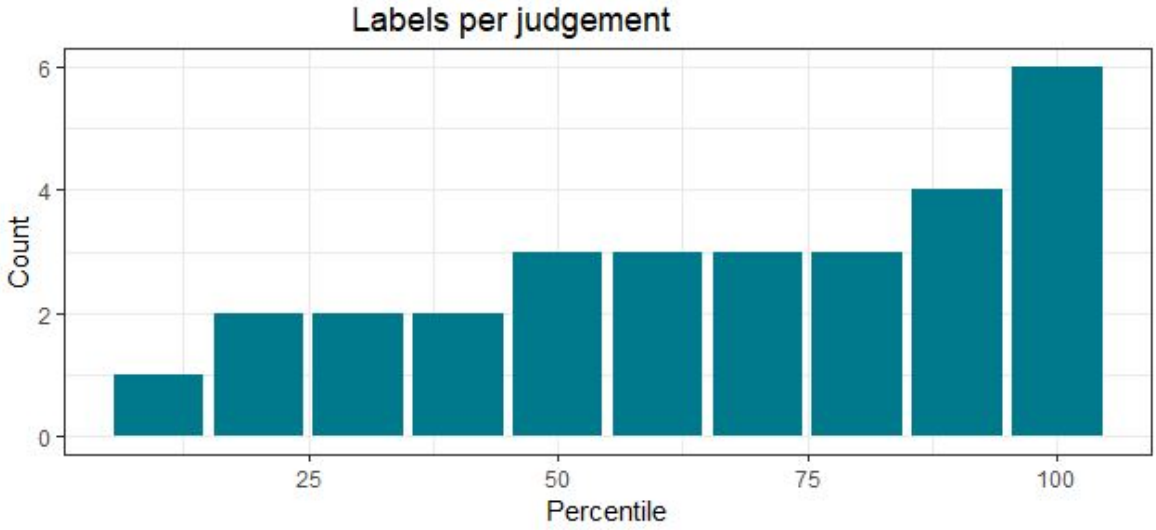

Figure 3.2: Labels per judgement across corpus

As observed in the figure 3.2 above, the number of labels assigned per judgement is less than equal to just 3 labels for up to 80% of all the documents provided and about 4 labels for another 16% of the documents. Totally making about 96% of the judgements containing less than 4 labels. The minimum of the labels assigned is 1 and maximum of 9 with an average of 2.7 labels per document. However, selecting the number of recommendations to generate is optional and it is clear that just providing 3 recommendations will be able to justify 80% of the judgements if the distribution remains the same in the future documents.

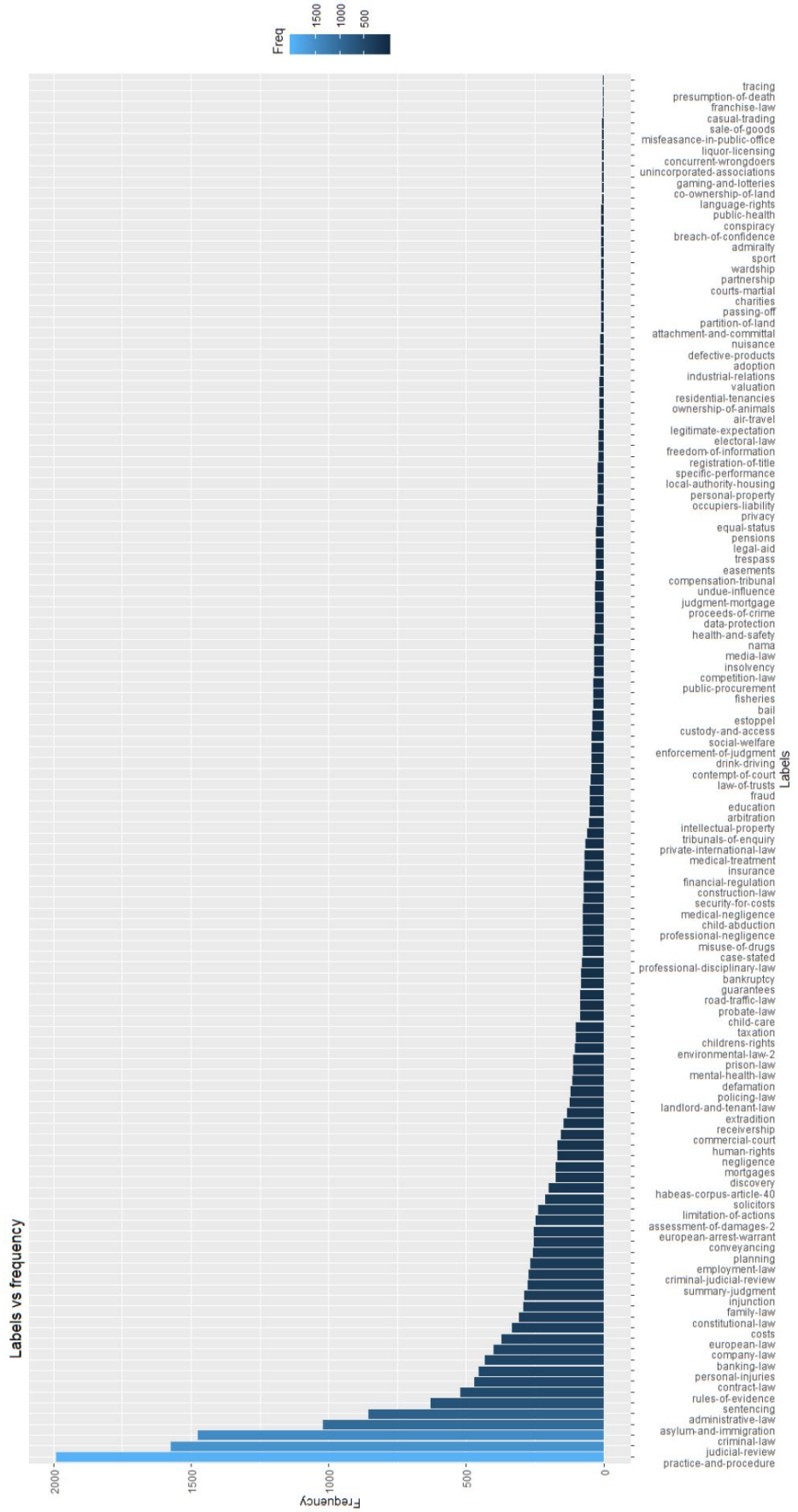

Figure 3.3: Frequency of labels

3.1.2 Scraped text data

Word embeddings require an enormous amount of text data to develop and the utilization of judgements data from the previous section will provide an unfair advantage to the models to know patters prior new appearance and exhibit the behaviour of overfitting. Hence data apart from the above judgements must be procured to develop word embeddings to avoid overfitting. The data provided from State Decisis was about 6891 judgements and the source `www.courts.ie` contains about 13000+ judgements as of 20th October 2019. Firstly, all of the available judgements from this source was scraped. Next, the JSON file provided by Stare Decisis also contained a column called *Link* and these links were used to identify the judgements that are present in both the scraped and initially provided. Once identified these judgements were removed from the scraped judgements. There were about 6100 judgements available for the development of embeddings model. The incorporation of more text for embeddings space will only help improve the quality of word vectors as suggested by (Mikolov, Sutskever, et al., 2013). Hence, more legal data was scraped from https://www.supremecourt.gov/oral_arguments/argument_transcript/ which is the United States of America website containing transcripts of oral arguments from Supreme court and the records available were from the year 1968 to 2019 with the total of about 5000 text documents. Finally, there are about 11000 legal text documents in total for the development of word embedding. Basic pre-processing steps were applied on these before proceeding:

• Parsing the HTML to extract the judgement text (For Irish www.courts.ie judgements only)

• Remove all escape characters (such as , ,̊), punctuation, single character words, extra white spaces.

• Convert the text to lowercase.

• Tokenize while maintaining the order of the words.

The below table provides statistics on these after pre-processing:

Vocabulary length	270,398
Vocabulary intersection with labelled judgements	73.5%
Total words	≈64 Million

Table 3.3: Scraped document statistics

The total number of words present in scraped data is about 64 million. Its vocabulary size is several times bigger than the judgment text received from *Stare Decisis*. However, the intersection of the vocabulary between them is only about 73.5%. The decision must be made about the missing vocabulary during the phase of the experiment and more on this will be discussed in detail appropriately for each approach later in this chapter.

3.2 Research Design overview

This section will provide the complete blueprint of how the research is designed to solve the problem of labelling Irish legal judgements. Multiple experiments will be performed using traditional and state-of-the-art techniques and these models will be evaluated using precision@N, recall@N and F1@N. F1@N will only be used to compare during the experiments as it provides a single number score involving both recall and precision.

The research aims at a detailed exploration of the recommender performance with each incremental changes made with either the feature representation or the similarity measure. However, several aspects revolve around the approach that can be further modified and tested such as the vector length of word embedding, negative sampling, number of epochs to learn word vector, window size and so on which requires tremendous computation and time. The efforts are only made to study the impact of the changes made in the approach rather than the embedding model fine-tuning. Default settings are used with all the models developed and they are very similar to Google's pre-trained embedding.

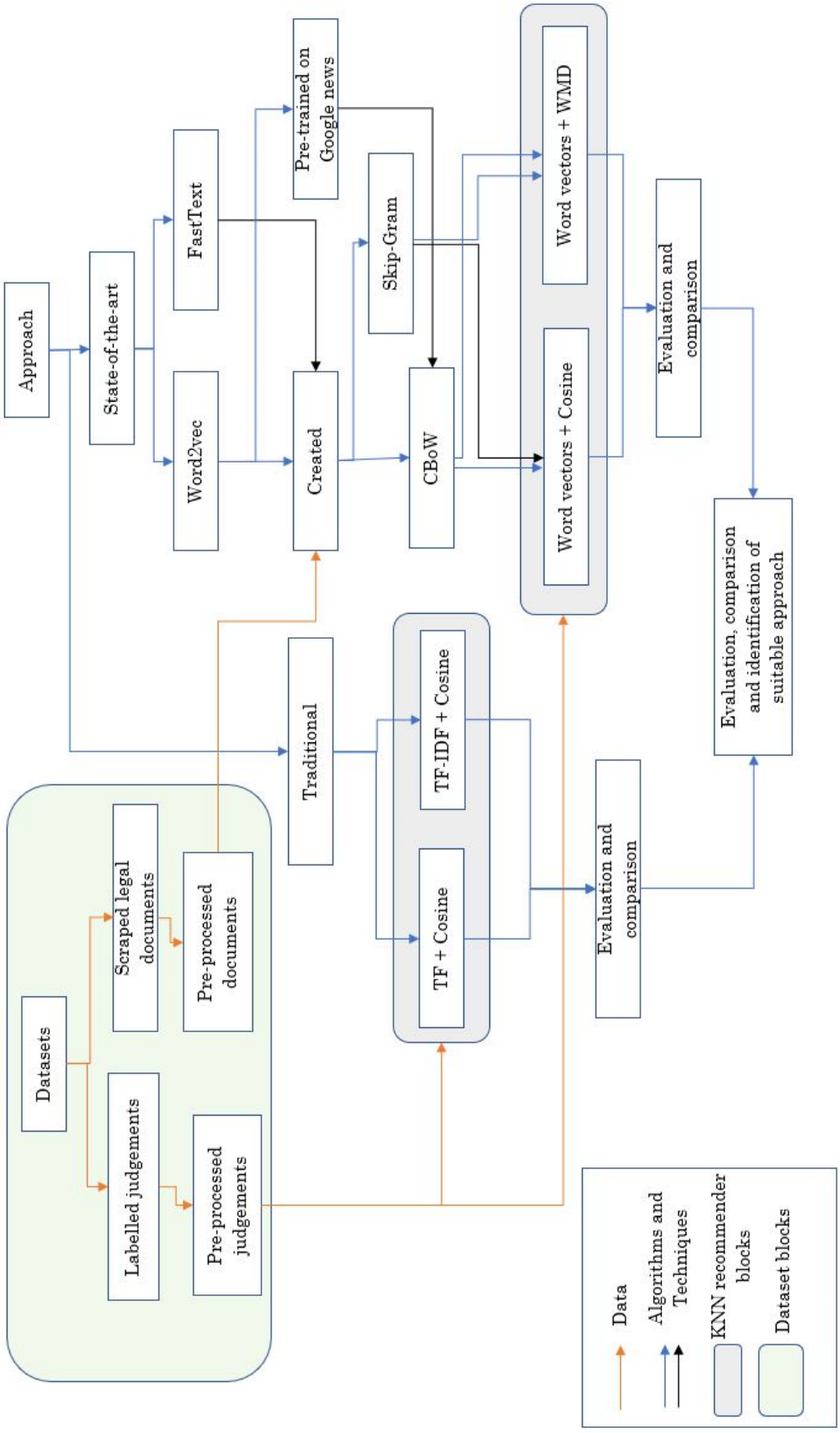

Figure 3.4: Overview of research design

The figure 3.4 shows the complete workflow of this research. There are 2 segments to it namely, the dataset section which is highlighted on the top left corner and the functional segment which is everything else connected apart from it.
The dataset block contains two datasets at the start, labelled judgements provided by Stare Decisis and scraped legal documents from Irish and USA legal websites. Both the datasets are pre-processed accordingly and there are subtle differences in their steps and are mentioned in the previous section. Pre-processed labelled judgements are used to develop and test traditional TF and TF-IDF with KNN approach with cosine similarity.

The scraped legal documents after pre-processing are provided to word2vec and Fast-Text algorithms to develop word embedding. There are again two types of architecture to develop word embedding CBoW and Skip-gram. With word embeddings, both cosine and word mover's distance can be applied to identify the similarity between the documents. Now, the experiments are designed in such a way that comparisons between the models developed are made optimally before identification and conclusion of the best performing approach among the approaches chosen as part of this research. There are four key approaches chosen to arrive at the recommendations using KNN recommender in this research. They are,

- TF vectors as feature representation and cosine similarity
- TF-IDF vectors as feature representation and cosine similarity
- Word vector-based feature representation and cosine similarity
- Word vector-based feature representation and word mover's distance

The same is reflected in the overview diagram provided above. Word vector covers all the algorithms, architectures and training methods used.

3.3 Research methodology

The methodology of this research, irrespective of the approach used is no different than supervised learning. The components on the higher level are the same.

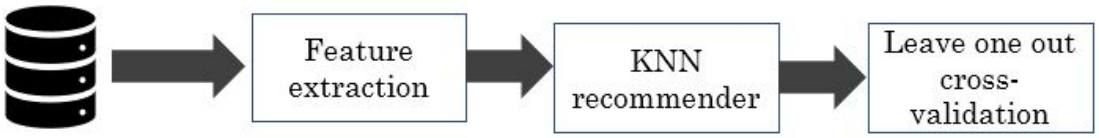

Figure 3.5: Research methodology

Note: The cosine similarity matrix is pre-computed during each approach to speed up the process of evaluation and this does not violate any condition of including training data point.

Instead of drastically running over to the most efficient approach as found in the literature this research moves quietly with a simpler and gradual increment from approach to approach slowly growing towards incorporating the state-of-the-art components in a search to identify the best recommender for Irish legal judgements.

The baseline starts at using TF vectors and cosine similarity and steps are taken to explore and grow towards the embeddings approach to find a similarity between the documents to recommend labels.

3.4 Recommender types

3.4.1 Type 1 - TF vectors and cosine

Type 1 recommender will provide with the baseline score to look up for and improvement will be attempted from thereon. The pre-processing steps are already performed at this point and clean data is used both concerning labelled judgements and scraped data.

The below are the steps taken to develop this recommender:

1. Extract TF vectors from the data for each documents.

2. Compute cosine similarity between each document using TF vectors and construct a similarity matrix

3. For each document identify similar documents and rank them in non-increasing order.

4. Perform leave one out cross-validation with range of K values and plot the recall@N, precision@N and F1@N scores.

5. Identify the optimal K value using precision@N vs recall@N plot.

6. Generate charts accordingly for 3,4 and 5 recommendations per judgement to show the effect of varying number of recommendations.

3.4.2 Type 2 - TF-IDF vectors and cosine

Type 2 recommender will encompass an incremental change over the previous one. Here TF-IDF vectors are used instead of TF vectors and rest of the process remains the same as before. As per the literature this type of recommender is widely used to perform text analytics.

3.4.3 Type 3 - Resultant word vectors and cosine

Type 3 turns a completely different direction as compared to previous ones. Word embedding algorithms are used to learn the numerical representations of the words that capture both the semantic and syntactic relationship in a meaningful way.

Two of the popular algorithms word2vec and FastText are used to develop word embedding. Also, word2vec pre-trained on Google news dataset will be used as a starting point for embeddings and further exploration will be done considering this as the baseline for word-embedding based recommenders.

Both word2vec and FastText provides word vectors which are the numerical repres-

entation for the words in the document. To extend this to compare documents vector addition is performed to add word vectors to retrieve document vector.

To handle missing words word2vec algorithm will not have a solution for the words not in vocabulary and they have to be removed from the documents before processing. To solve this problem FastText is used as it operates at the level of character-level n-grams unless any sequence of characters present in the out of vocabulary words are present as n-grams with vectors then word vector will be generated irrespective of its presence in the model's vocabulary. The principle is similar to what (Le & Mikolov, 2014) have demonstrated to perform word vector addition for context multiplication resulting in meaningful representation.

Embeddings plus cosine:

1. Develop word embedding using scraped data that is completely independent of the labelled judgements received from Stare Decisis.
2. Extract word vector for each word in labelled judgements and calculate resultant vector to represent the document.
3. Compute similarity between each documents using vectors representing context of the documents.
4. For each document identify similar documents and rank them in non-increasing order.
5. Perform leave-one out cross-validation with range of K values and plot the recall@N, precision@N and F1@N scores.
6. Identify the optimal K value using precision@N, recall@N and precision@N vs recall@N charts.
7. Generate charts accordingly for 3,4 and 5 recommendations per judgement to show the effect of varying number of recommendations.

In step 1, both word2vec (created and pre-trained) and FastText along with CBoW and skip-gram architectures are used to develop word embedding separately and all the other steps are performed independently for each.

3.4.4 Type 4 - Word vectors and WMD

In this type word mover's distance(WMD) is used instead of cosine similarity. The advantage of using WMD over cosine is the distance metric itself is designed in such a way that it uses word vectors for each document and estimates the effort required to transport words from one document to other and hence there is no need to aggregate word vectors manually to represent documents.

Embeddings plus word mover's distance:

1. Develop word embedding using scraped data that is completely independent of the labelled judgements received from Stare Decisis.
2. Extract word vector for each word in labelled judgements.
3. Use word vectors of each documents to calculate similarity matrix using WMD.
4. For each document identify similar documents and rank them in increasing order. WMD is a distance metric and numerical value higher means higher difference between the documents.
5. Perform leave one out cross-validation with range of K values and plot the recall@N, precision@N and F1@N charts.
6. Identify optimal K value using the charts generated.
7. Generate charts accordingly for 3,4 and 5 recommendations per judgement to show the effect of varying number of recommendations.

Again, in step 1 both word2vec(created and pre-trained) and FastText are used to develop word embedding separately and all the other steps are performed independently for each. This type is considered state-of-the-art according to the literature.

3.5 Evaluation

Leave one out cross-validation(LOOCV) is used to evaluate the models developed. Since all of the observations from the dataset will be used to evaluate, the results calculated will be far less biased. Meaning the split made for training and testing

from other evaluation methods could not have the same distribution of labels. If the test label distribution was different than the training label distribution, then the model developed would be more biased towards the distribution found in the training and the metrics calculated would not be fair when evaluating on the test set.

LOOCV is generally very expensive with high computational complexity, but it is feasible in this research as the dataset to evaluate contains just 4000+ judgements to evaluate.

Regarding the metrics used to evaluate Precision@N, Recall@N and F1@N are used while F1@N score is solely used to compare any two recommenders as it provides a single score involving both precision@N and recall@N.

Precision-Recall is used to measure the success of predictions with classification and recommendation tasks when the outcome variable is very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned (Pedregosa et al., 2011).

The precision-recall curve is designed to understand the trade-off between precision and recall for different thresholds. The threshold in this research means several recommendations provided. A high area under the curve represents both high recall and high precision, where high precision relates to low FP, and high recall relates to low FN. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. An ideal system with high precision and high recall will return many results, with all results being relevant.

Precision@N, recall@N and precision-recall curves will be used to evaluate the performance of each recommender developed. Precision@N, recall@N and F1@N will be used interchangeably in this research as precision, recall and F1 respectively.

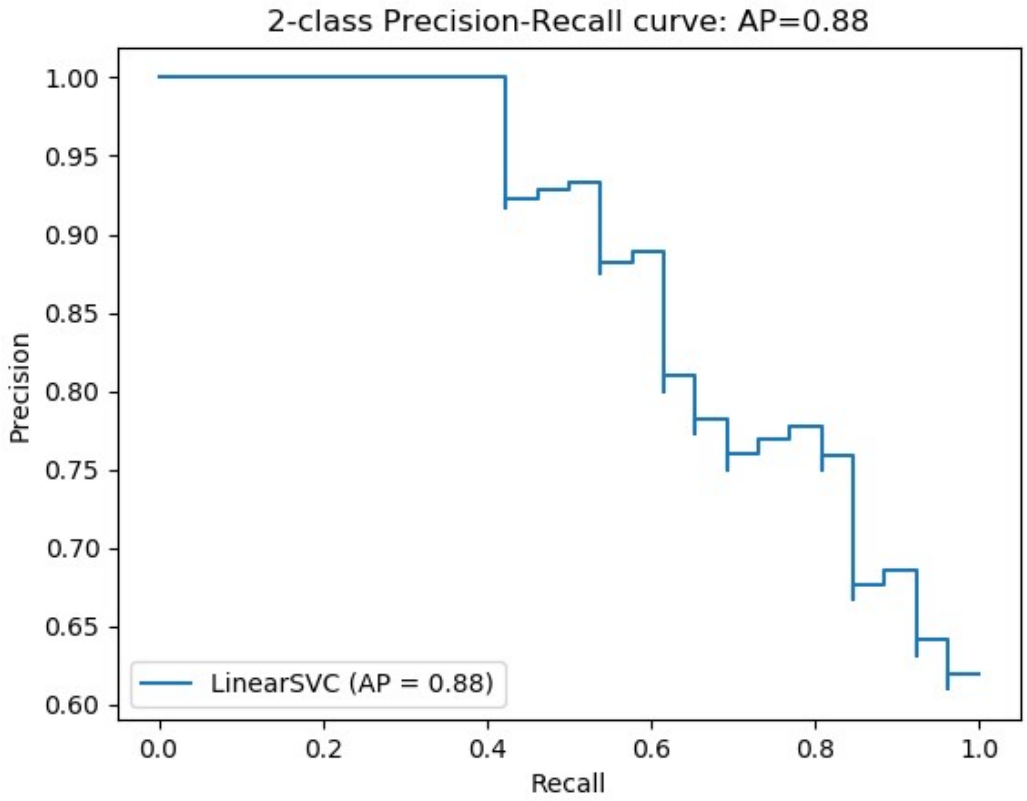

Figure 3.6: Precision vs Recall curve

(Source: `https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html`)

3.6 Summary

All the details of the datasets were provided in the first section including the pre-processing done appropriately for both the datasets. A complete overview of how the research design was provided connecting the datasets and techniques in consideration for this research. Later research methodology provided a higher-level understanding of how the datasets are utilized to generate recommendations and later evaluation. Recommender types section provided an in-depth explanation with steps for each recommender and the motto behind each. Evaluation is the key to any research and justification is provided to both the method and metrics of evaluation used in this research.

Chapter 4

Evaluation

Four different types of recommenders have been outlined in the previous chapter and they will be used in this chapter in the experiment sections. Implementation section under experiments shows how each recommender type has been implemented and compared. Experiment summary section under experiments provides a brief explanation about the results obtained and knowledge gained from the experiment.
Each experiment is performed with a question to answer while progressing towards solving the research problem. Experiment 1 is the starting point for this research and based on the results observed there further experiments will be designed accordingly. Leave one out cross-validation method is used throughout and will not be mentioned explicitly in the implementation diagrams.

Table 4.1 provides the details of all the recommenders that are built as part of this research. In table 4.2 TimeD is the amount of time taken to develop word embedding and TimeR is the amount of time taken to extract word vectors and calculate resultant vector for each document. Vocabulary intersection is the percentage of words present in both judgement text and words learnt by model. The vocabulary intersection initially was about 73% and after the development of word embeddings it changes due to negative sampling performed by the algorithm during the development.

Index	Text representation	KNN measure	Embedding type	Embedding architecture
1	TF	Cosine	-	-
2	TF-IDF	Cosine	-	-
3	Resultant vectors	Cosine	Domain specific(word2vec)	CBoW
4	Resultant vectors	Cosine	Domain specific(word2vec)	Skip-gram
5	Resultant vectors	Cosine	Pre-trained (word2vec)	CBoW
6	Resultant vectors	Cosine	Domain specific (FastText)	CBoW
7	Resultant vectors	Cosine	Domain specific(FastText)	Skip-gram
8	Word vectors	WMD	Pre-trained (word2vec)	CBoW
9	Word vectors	WMD	Domain specific(word2vec)	Skip-gram
10	Word vectors	WMD	Domain specific (FastText)	CBoW
11	Word vectors	WMD	Domain specific(FastText)	Skip-gram

Table 4.1: Recommenders developed during this research

Algorithm	Architecture	Vocabulary intersection	TimeD	TimeR
Word2vec	CBoW	57.1%	28mins	48 mins
Word2vec	Skip-gram	57.1%	152 mins	51 mins
FastText	CBoW	62.8%	54 mins	64 mins
FastText	Skip-gram	62.8%	239 mins	65 mins

Table 4.2: Details of word embeddings developed

4.1 Experiment 1

4.1.1 Aim

To identify the best performing KNN recommender with cosine similarity between TF and TF-IDF text representations when both are evaluated using LOOCV method with precision, recall and F1 metrics.

4.1.2 Implementation

Dataset used: Pre-processed version of the labelled dataset provided by *Stare Decisis*.

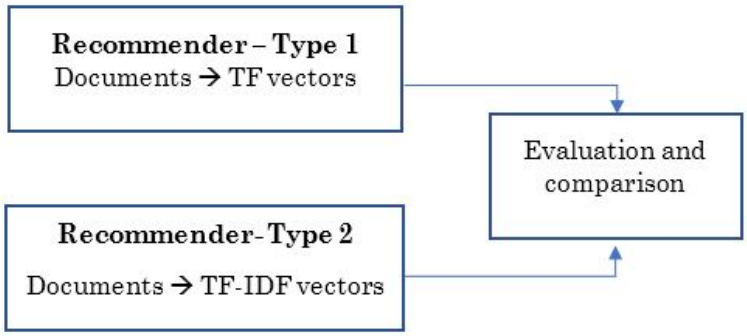

Figure 4.1: Experiment 1: Implementation

Recommender Type 1: KNN recommender using cosine as similarity and TF vectors used to represent legal judgements.

Recommender Type 2: KNN recommender using cosine as similarity and TF-IDF vectors are used to represent legal judgements.

Evaluation and comparison: Leave one out cross validation is performed on each recommender to calculate evaluation metrics. Charts will be generated accordingly to support the comparison of the recommenders and F1 score is explicitly used for this purpose.

Only this experiment is performed using *R programming* environment and the configuration of the computing device and time taken to develop are given below.

Computation device configuration:

Processor: Intel core- i7 - 8550U

RAM: 16GB

On an average time taken to,

extract TF /TF-IDF vectors $\approx$ 20 mins

calculate similarity matrix TF /TF-IDF $\approx$ 9 hours using *tm* package.

Generate recommendations ≈ 20 mins

4.1.3 Results

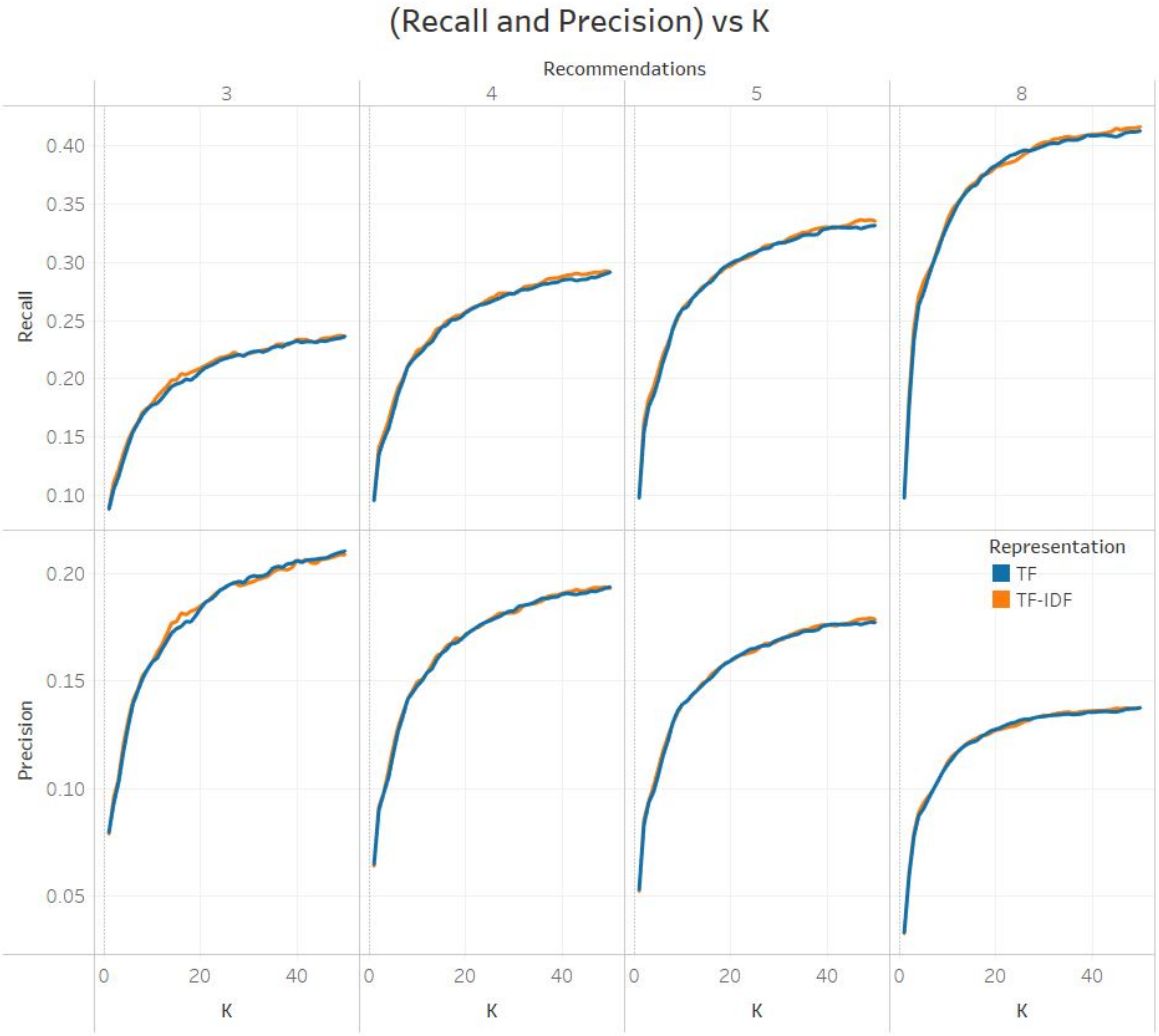

Figure 4.2: Experiment 1: (Recall and Precision) vs K

4.1.4 Experiment summary

Observing the recommendation results, both TF and TF-IDF vectors were almost similar. In terms of performance neither of them is worth exploring any further as the maximum of recall and precision for both are below *0.25* at 3 recommendations per judgement. F1 reflects the same as neither of them were able to go past *0.25* with any

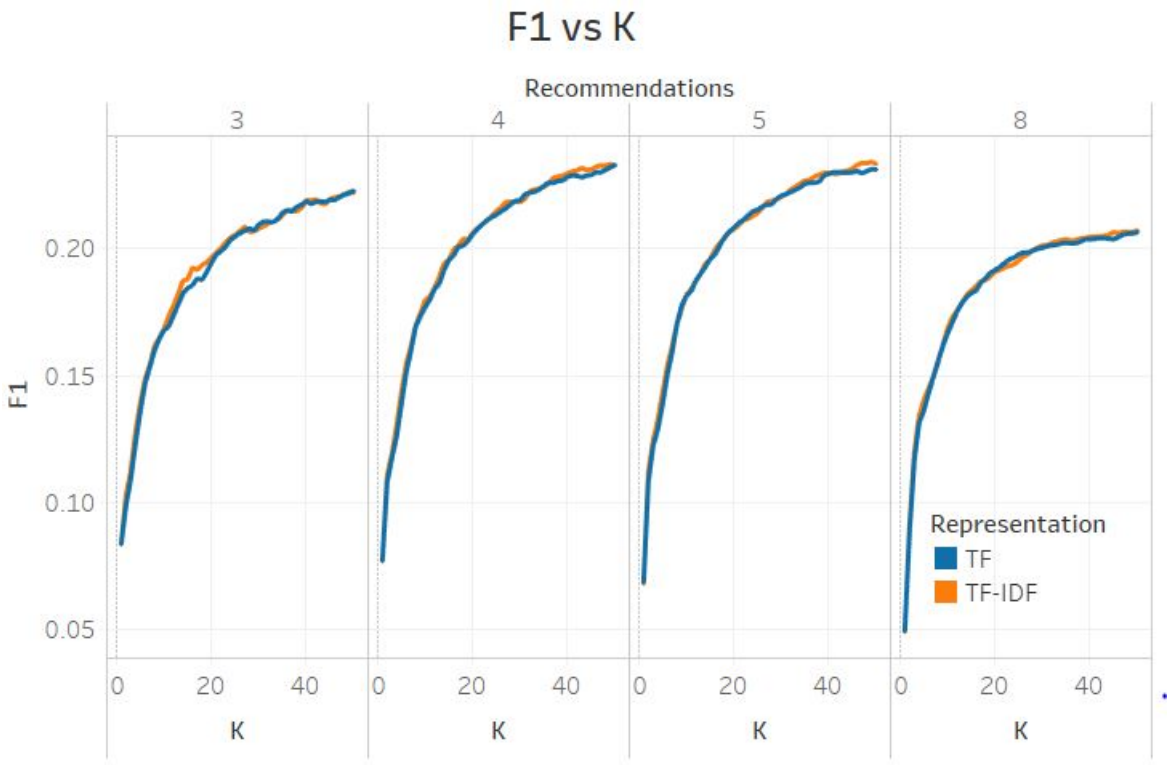

Figure 4.3: Experiment 1: F1 vs K

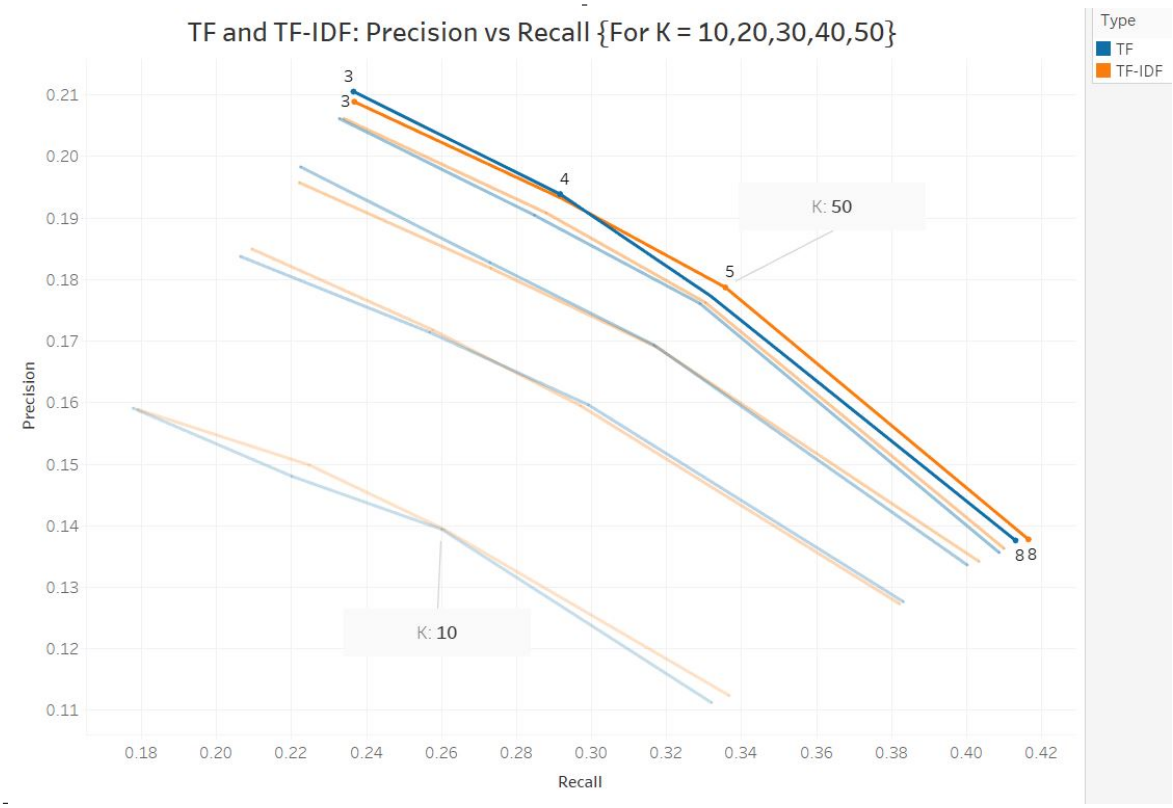

Figure 4.4: Experiment 1: Precision vs Recall

number of recommendations or K value.

Irrespective of the performance the charts show a general trend with respect to recall and precision over increase in number of recommendations. High recall results in lower recall and lower recall results in higher precision.

Thus, precision vs recall chart is developed to investigate more on this precision and recall see saw effect, and it clearly shows a trend that lower number of recommendations provides higher precision and increasing the number of recommendations will have a higher recall. A balance between the two as observed is around 5 recommendations. This chart also suggests that increasing the number of K for KNN also has an effect on precision and recall. The increase in precision and recall is significant with lower values of K but saturates as K approaches 50.

4.2 Experiment 2

The previous experiment has shown that both the traditional feature representation have provided poor results. The focus from here on will be on word embedding techniques to solve the research problem.

4.2.1 Aim

The aim of this experiment is to use the word2vec algorithm-based pre-trained embedding and domain specific embeddings developed with both CBoW and skip-gram architecture to calculate resultant vectors for document representation to explore and compare their effectiveness in recommending labels for Irish legal documents when KNN recommender approach is used and all of the recommenders developed are evaluated using leave one out cross validation method with precision, recall and F1 metrics.

4.2.2 Implementation

All of the recommender blocks used in this experiment are of the same type on the surface. From inside, the difference varies with model development or the architec-

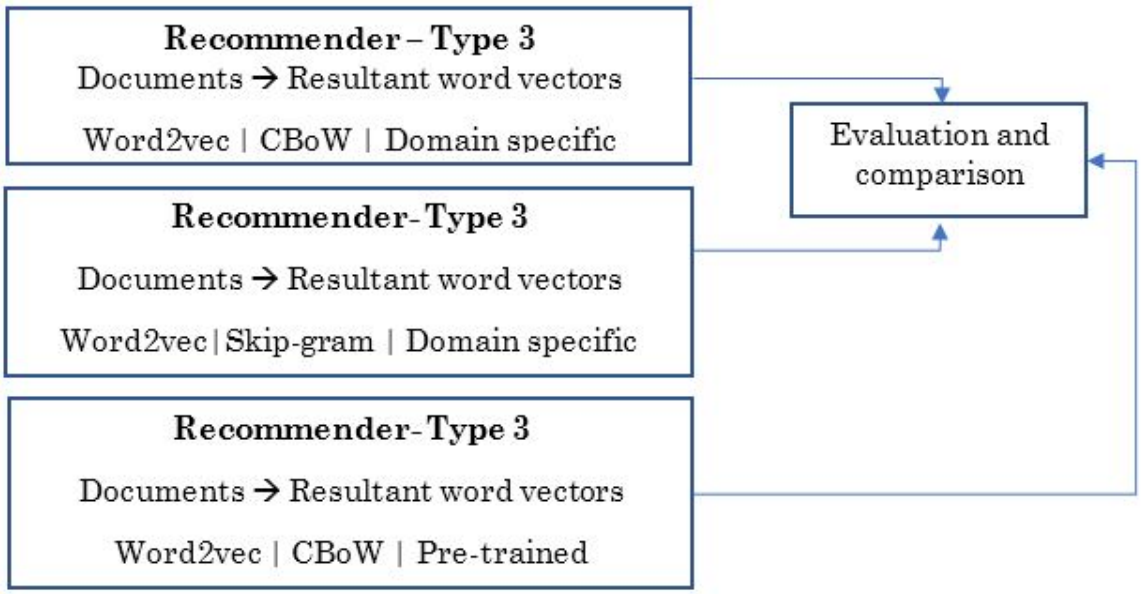

Figure 4.5: Experiment 2: Implementation

ture. The first recommender block is the CBoW architecture of word2vec and the embedding are developed using domain-specific data scraped as explained in chapter 3. The second recommender block is the same as the first but only with the change in architecture used. Skip-gram is used in this block instead of CBoW. In the final block, word2vec word embedding pre-trained on Google news data is acquired and used which is of CBoW architecture.

The vocabulary intersection between the pre-trained embeddings and the labelled judgements is about 46.1% while the word2vec embedding developed and the judgements are about 57.1%. The number of words used in pre-trained embeddings is 100 Billion words while the developed embedding has just 60 million words.

4.2.3 Results

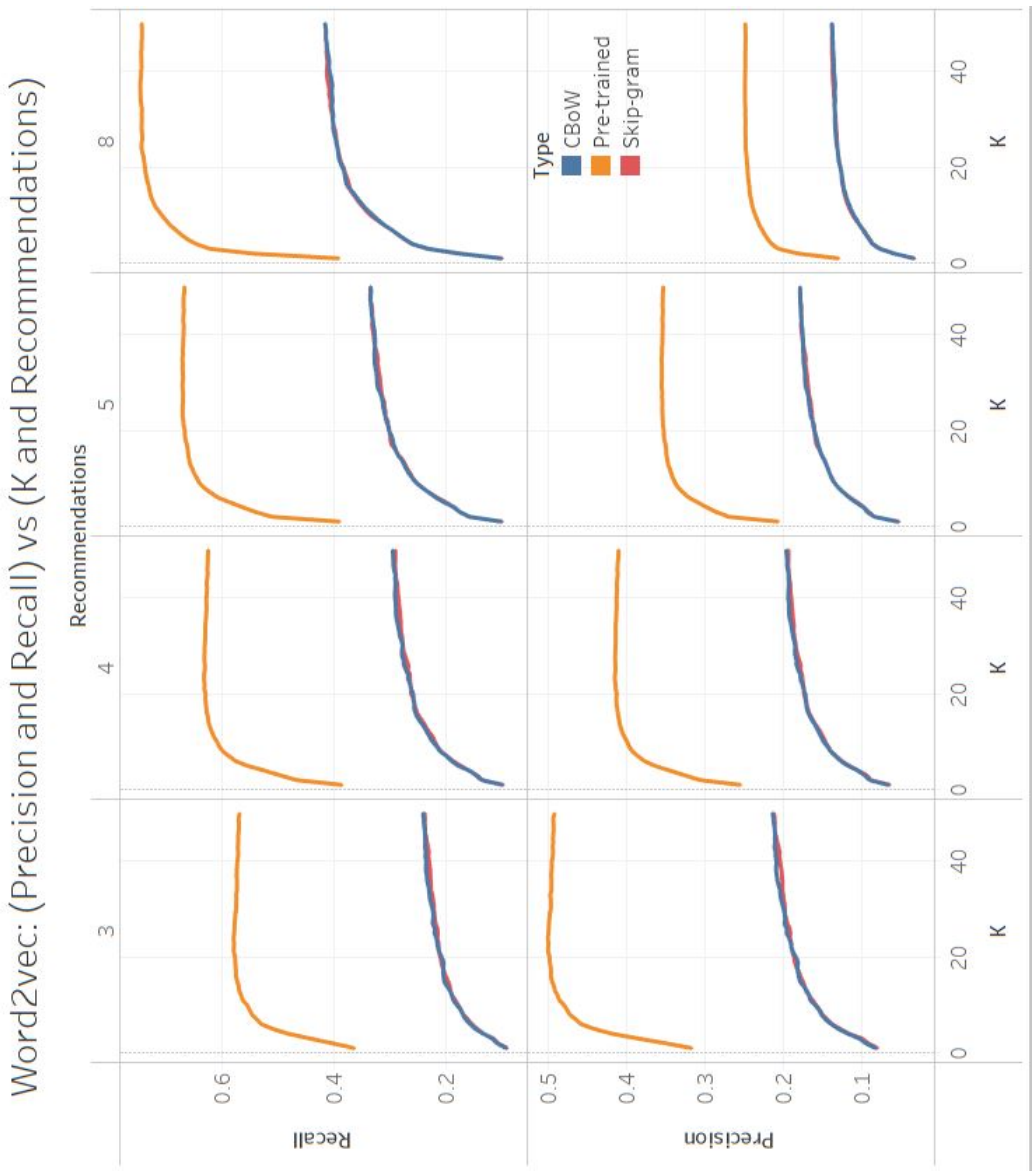

Figure 4.6: Experiment 2: (Precision and Recall) vs (K and recommendations)

Figure 4.7: Experiment 2: F1 vs K

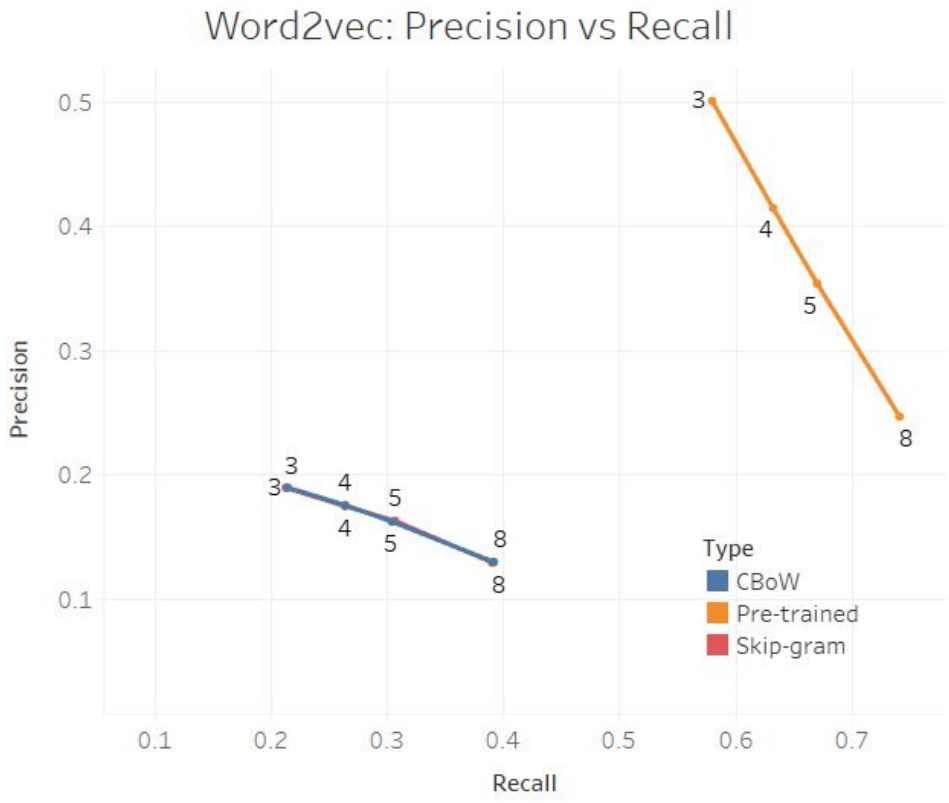

Figure 4.8: Experiment 2: Precision vs Recall

4.2.4 Experiment summary

This experiment has provided valuable insight into how the number of words used to develop word embedding could affect the performance of the model. Even though the vocabulary intersection was greater with the embedding developed the results suggest that the number of words could be the key with word2vec embedding, both CBoW and skip-gram architecture.

In terms of numbers, pre-trained embeddings outperformed developed embeddings in every possible way. Precision and recall charts show a similar trend as the previous experiment concerning the increase in the number of recommendations. Recall increased with an increase in the number of recommendations while precision decreased with the same. The highest precision was achieved by pre-trained embedding with around 0.5 for $K = 24$ and the number of recommendations set to 3. The performance of all the methods saturated at around 18-24 nearest neighbours and started decreasing afterwards. Precision vs recall chart confirmed that there is a significant performance gap between developed embeddings and the pre-trained.

Having seen the performance and setting a benchmark of 0.5 precision and 0.53 F1 score for word2vec, FastText will be implemented in the next experiment for further exploration.

4.3 Experiment 3

The advantage of character-level embedding with FastText algorithm will have an effect in the form of increase in vocabulary intersection. It also provides better representation for infrequent words and also for out of vocabulary words as they will be composed of several n-grams in vector space.

4.3.1 Aim

The aim of this experiment is to implement FastText algorithm to develop word embedding and calculate resultant vectors to represent documents using both CBoW and skip-gram architecture to explore and compare their effectiveness in recommending labels for Irish legal documents when KNN recommender approach is used and all of the recommenders developed are evaluated using leave one out cross validation with precision, recall and F1 metrics.

4.3.2 Implementation

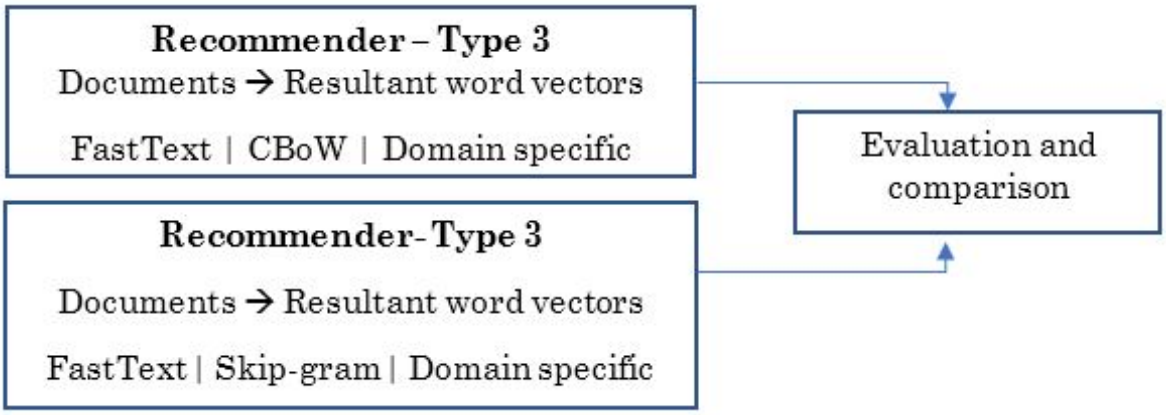

Figure 4.9: Experiment 3: Implementation

Both the recommenders used are very similar except for the architectural changes used to develop word embeddings. FastText uses character level n-grams to develop word embeddings over word2vec which completely deals with words.
Increase in vocabulary intersection was seen as expected with the judgements with about 62.8%.

4.3.3 Results

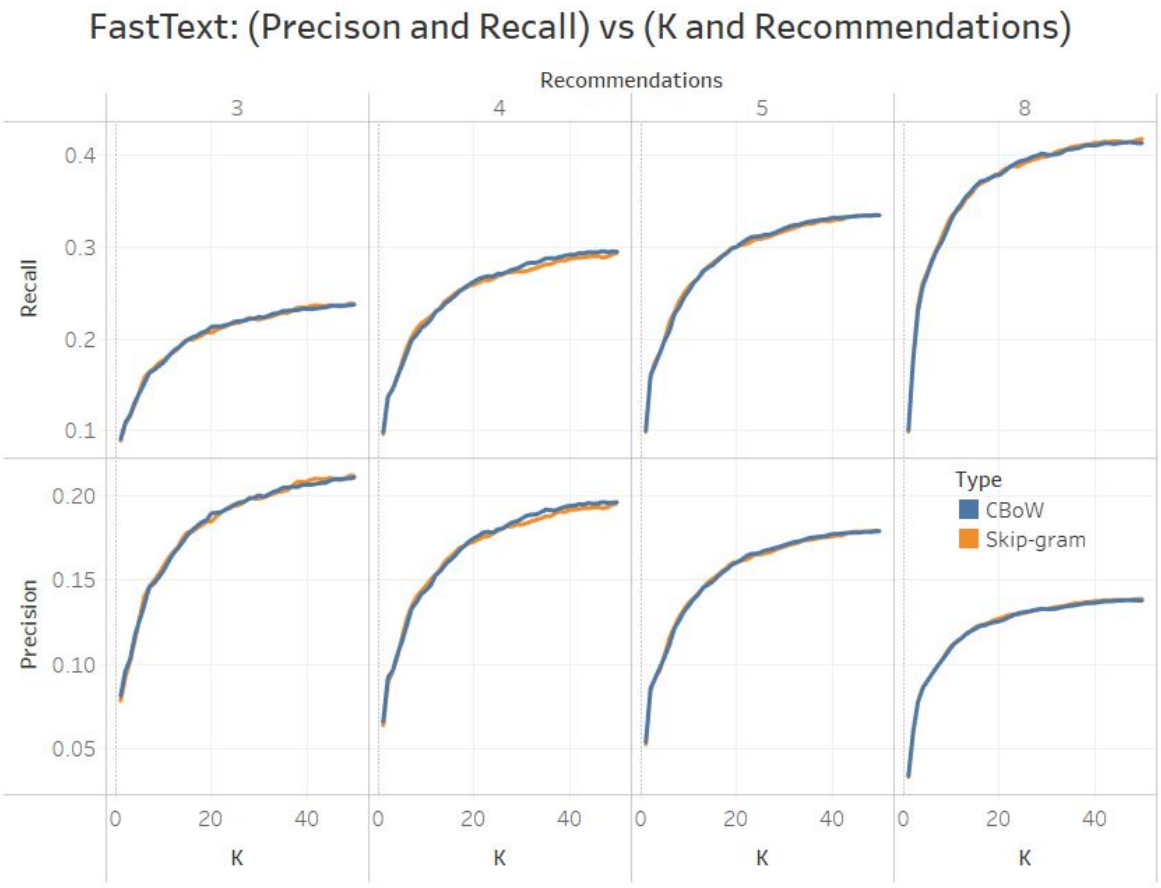

Figure 4.10: Experiment 3: (Precision and Recall) vs (K and Recommendations)

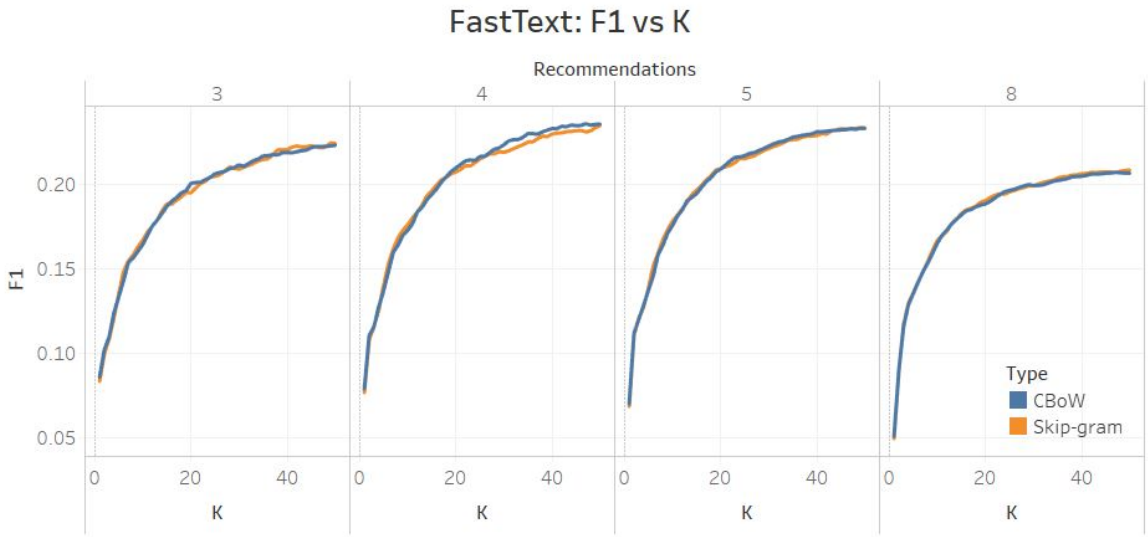

Figure 4.11: Experiment 3: F1 vs K

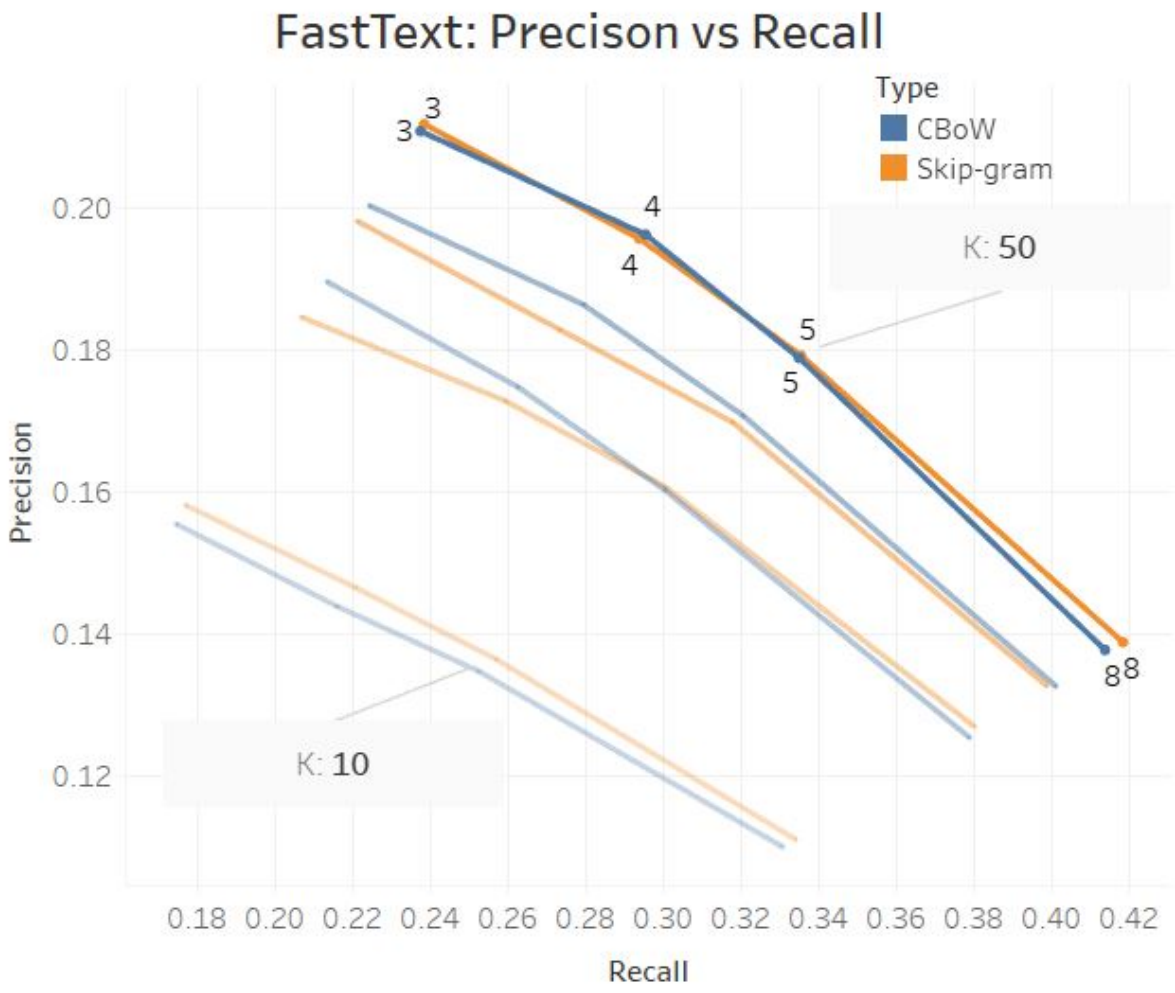

Figure 4.12: Experiment 3: Precision vs Recall

4.3.4 Experiment summary

The results show that there is little to no difference between word2vec and FastText developed embeddings. The use of cosine similarity in context with word embedding is only reliant on resultant vectors to identify the similarity between the documents using the context. In terms of the number they still appear poor and indifferent from the traditional vectors as observed in experiment 1.

4.4 Experiment 4

Domain-specific word embeddings fell short with both the word2vec and FastText algorithms when cosine similarity was used. The main reason so far appears to be the quality of the word vectors. WMD will be the final attempt at this problem in this research and it will be interesting if the results improve over cosine.

4.4.1 Aim

To identify among word2vec pretrained, word2vec domain specific and FastText domain specific the best performing word embedding in conjunction with the KNN recommender using WMD when all of them are evaluated using LOOCV with precision@N, recall@N and F1@N metrics.

4.4.2 Implementation

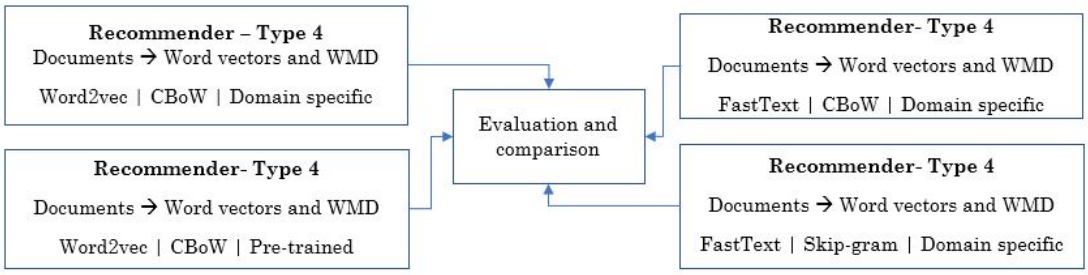

Figure 4.13: Experiment 4: Implementation

Since the computational cost of calculating WMD was very high and during experimentation, it was found that calculating WMD between two documents took on an average 70 seconds on a single GPU (Nvidia K80) using python 3.6 environment. By going with this number there are 4420 documents and the number of WMD calculations 4420^2 times 70 seconds. Since, WMD is commutative, meaning WMD(A,B) = WMD(B,A), it still requires $(\frac{4420^2}{2} - 4420) * 70$ seconds to calculate the distance matrix. Subtracting 4420, to avoid calculating document distance to itself. This number comes out to around 21 years to calculate the matrix. Hence, from a feasibility point

of view, a sample of 26 documents were chosen at random which still took about 6 to 7 hours per model to calculate the WMD matrix.

WMD for this sample chosen was calculated using FastText-Skipgram, FastText-CBoW, Word2vec-CBoW and Word2vec-Pretrained. The results of these are presented in the next section.

4.4.3 Results

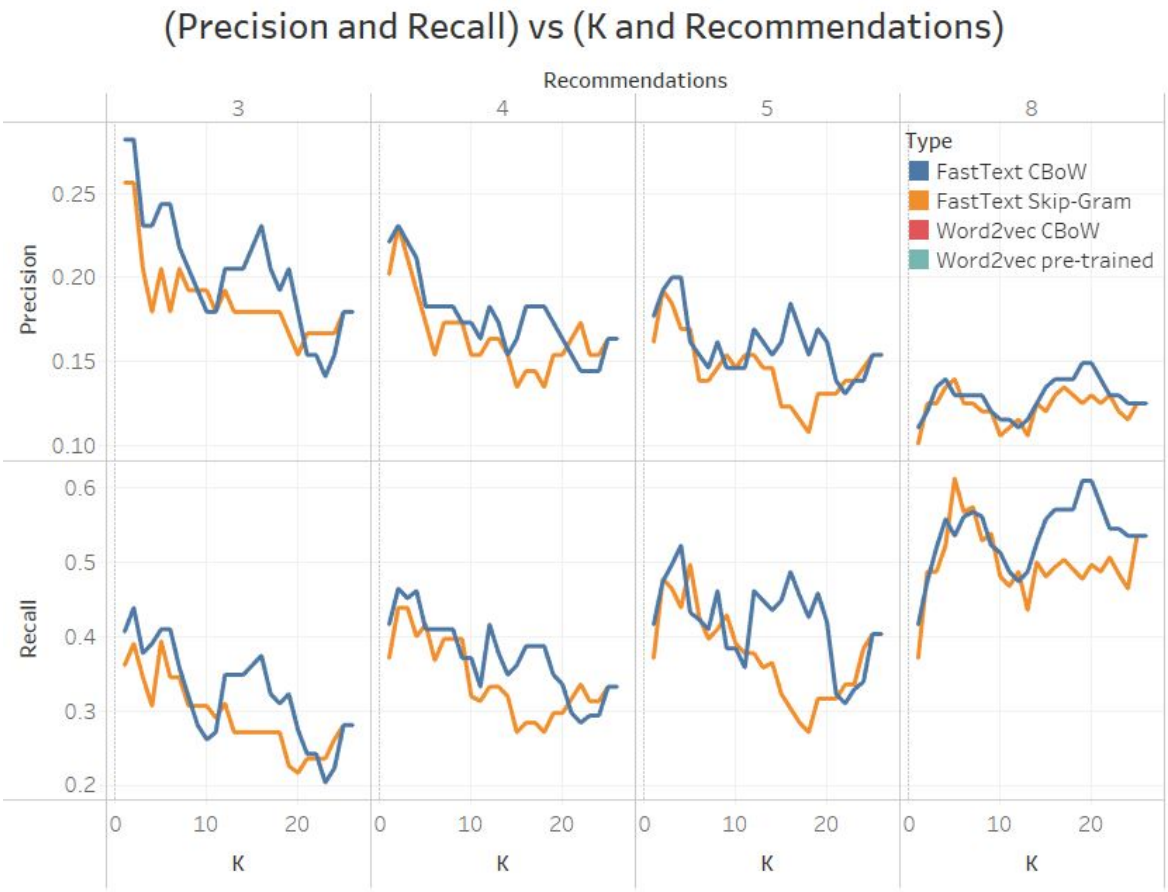

Figure 4.14: Experiment 4: (Precision and Recall) vs (K and recommendations)

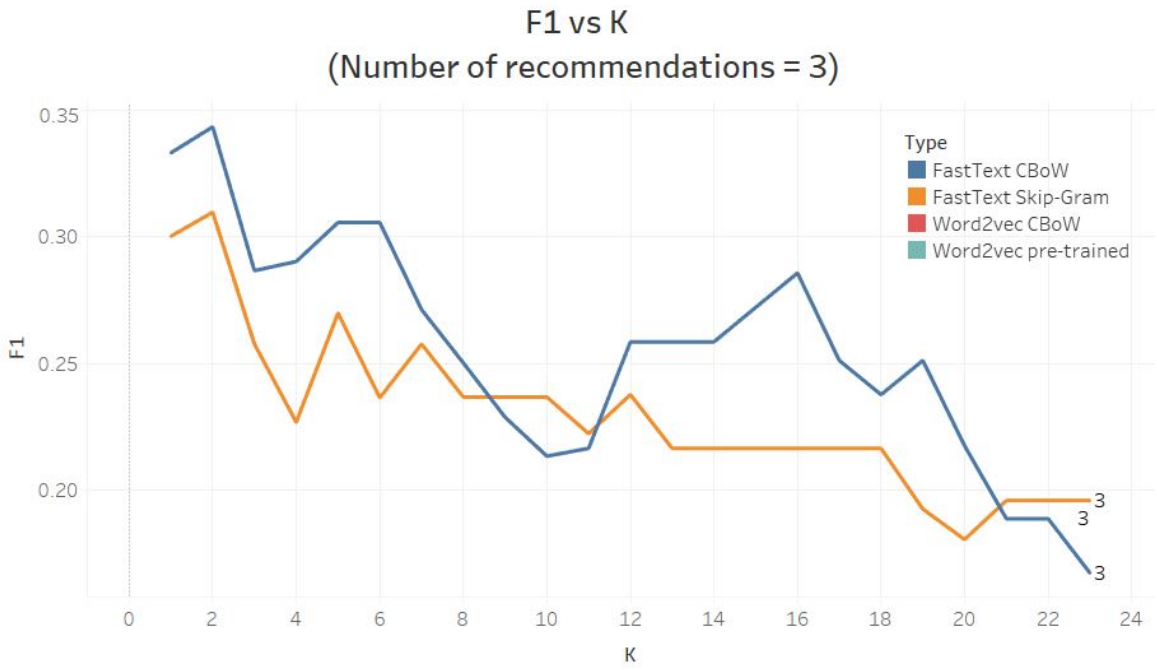

Figure 4.15: Experiment 4: F1 vs K

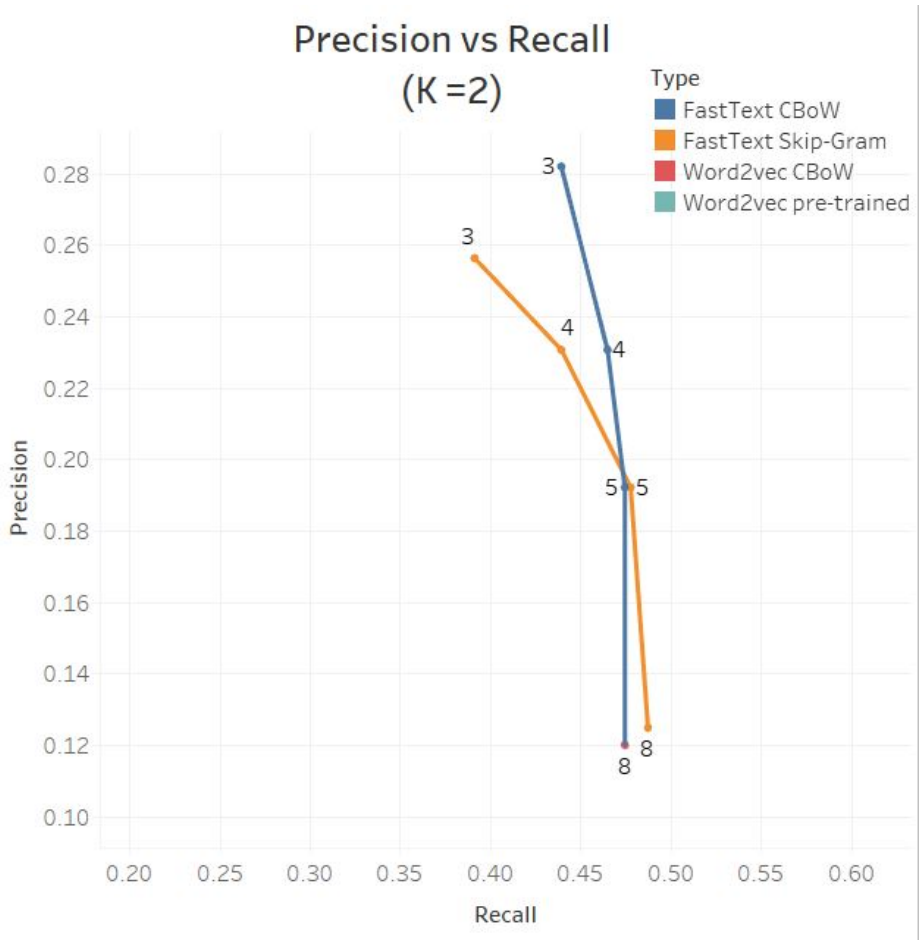

Figure 4.16: Experiment 4: Precision vs Recall

4.4.4 Experiment summary

The recommendations provided by word2vec CBoW and FastText CBow are exactly the same, and the same with word2vec pre-trained and FastText skip-gram. Hence there are only two lines visible throughout the graphs developed in this experiment as they are overlapping. Regarding the performance, WMD has provided precision of 0.28 and F1 of 0.34 with FastText CBoW and word2vec CBoW embeddings. To note, WMD only had 26 samples and the number appears to be at least 3% better than previous domain-specific embeddings. Pre-trained embedding has failed to deliver better results as it did with the cosine similarity approach. The small number of recommendations per judgement performed well even on this small dataset adhering to the definition of precision. About the selection of K WMD had a reverse trend as compared to cosine similarity. It appears throughout the charts that the lesser value of K resulted in better recommendation with K=2 being the sweet stop with about F1 = 0.34.

It would be interesting to compare FastText CBoW WMD approach which performed well in this experiment to pre-trained embedding with cosine which performed well in experiment 2 on the same sample dataset as WMD can not scale to the full dataset at this time. The results of the next experiment must be taken with a pinch of salt as the sample of the dataset used is very low.

4.5 Experiment 5

4.5.1 Aim

To identify the best performing KNN recommender between domain-specific FastText CBoW with WMD and pre-trained word2vec CBoW with cosine similarity when both are evaluated using LOOCV with precision@N, recall@N and F1@N metrics on a data sample of 26 judgements.

4.5.2 Implementation

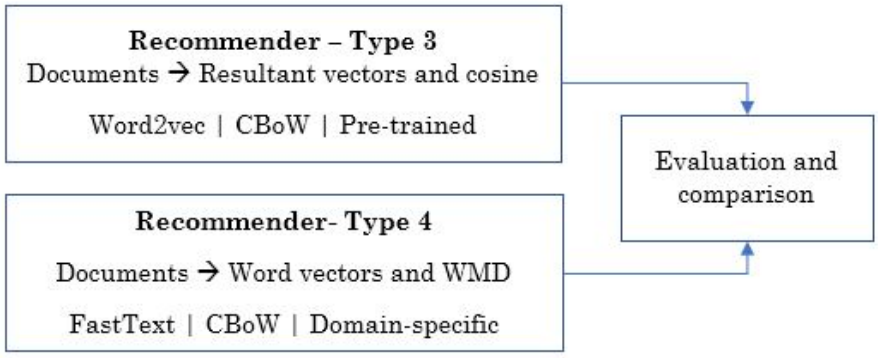

Figure 4.17: Experiment 5: Implementation

Best of both cosine and WMD recommenders identified in this research are compared on a sample data of 26 judgements. Domain-specific FastText CBoW embedding based recommender worked well with WMD while word2vec pre-trained embedding worked well with cosine similarity. Pre-trained embedding has established using resultant vector but failed to perform on WMD however, it is too early to say as further research must be conducted on the full data to identify a better performing approach.

4.5.3 Results

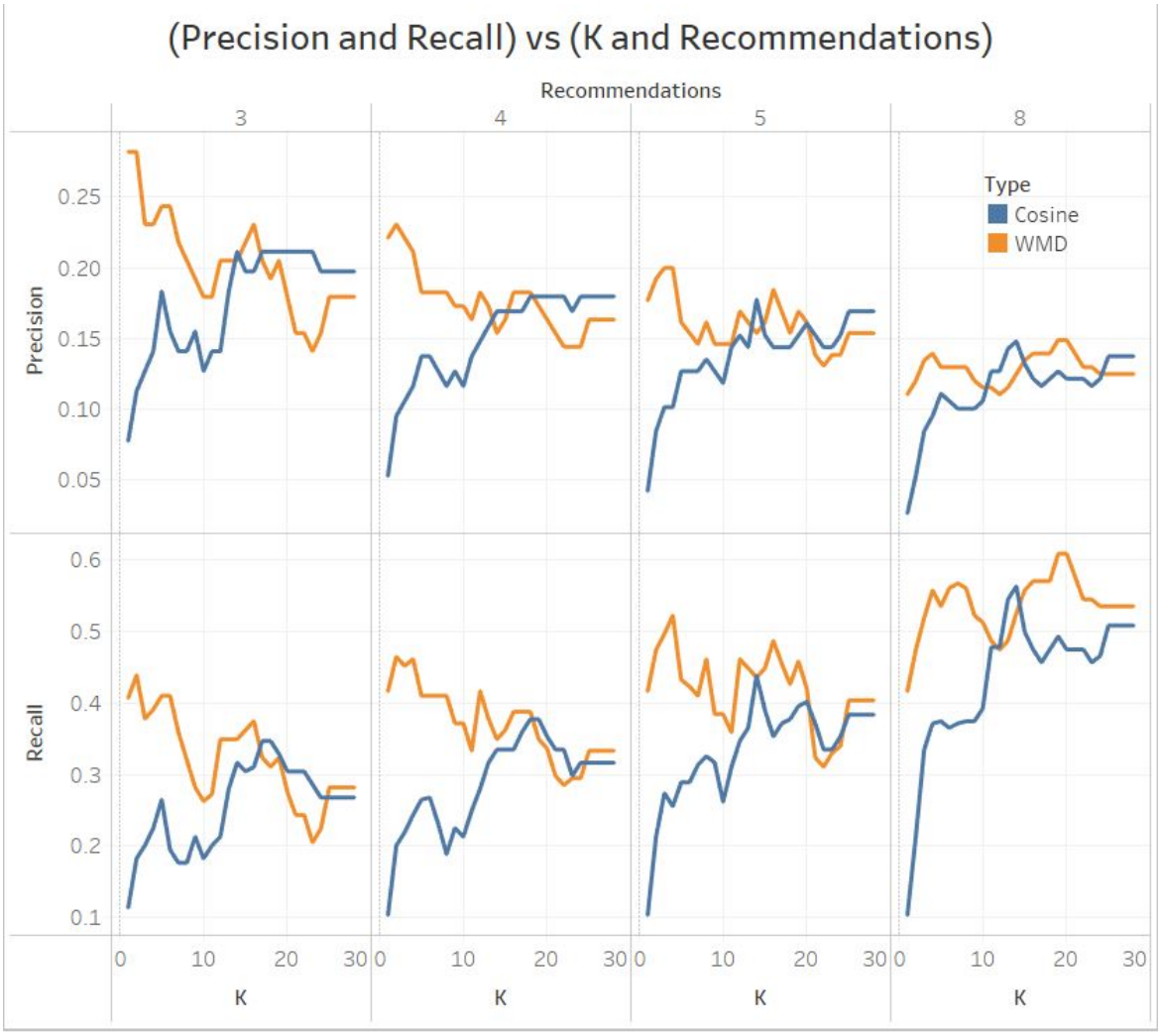

Figure 4.18: Experiment 5: (Precision and Recall) vs (K and recommendations)

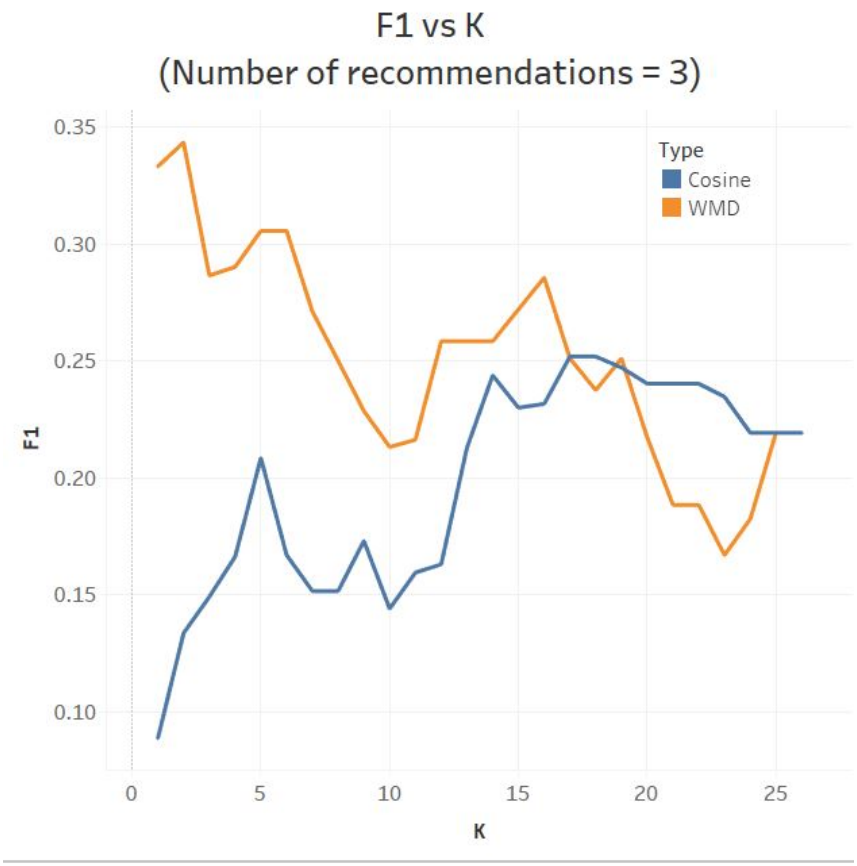

Figure 4.19: Experiment 5: F1 vs K

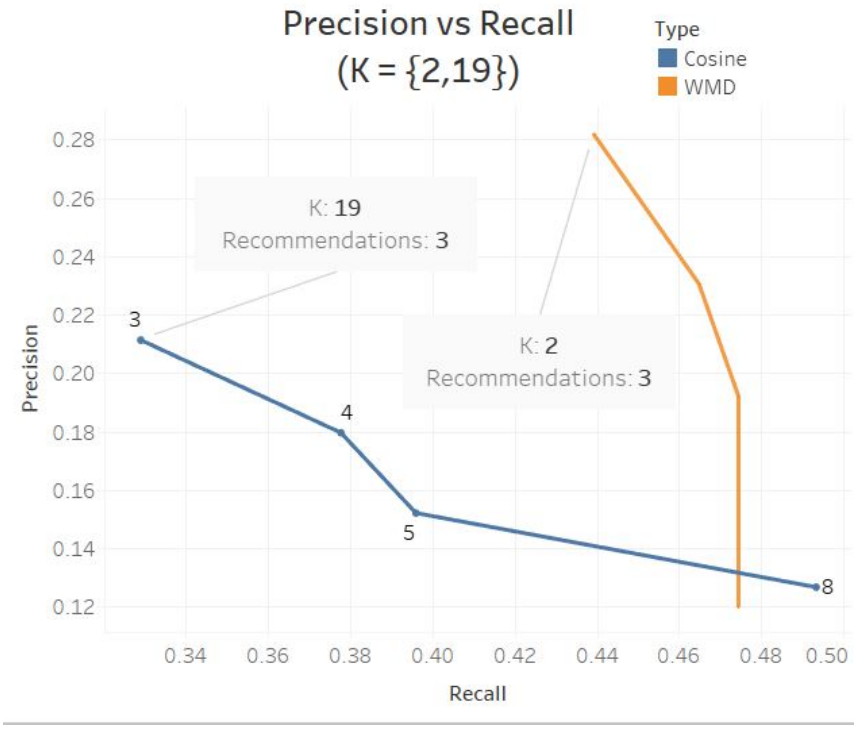

Figure 4.20: Experiment 5: Precision vs Recall

4.5.4 Experiment summary

The results obtained show that on the selected sample judgements of size 26 WMD approach still holds strong against cosine approach. The performance of WMD starts early as seen in the previous and this experiment and start dropping with an increase in K whereas with cosine it starts slow and increases with increase in K. This specifically handicaps cosine based approach as it requires sufficient nearest neighbours before the performance starts to saturate. Precision vs recall was plotted for both WMD and cosine approach with WMD read at 2 nearest neighbours and cosine read at 19 nearest neighbours each when they were highest according to figure 4.18 and figure 4.19. Both the approach have the highest values when the number of recommendation is low that is 3.

4.6 Discussion

Total of 5 experiments was conducted in this research to solve the labelling problem for Irish legal judgements.

Experiment 1 was conducted using traditional feature representations tf and tf-idf to identify a suitable one to solve the problem. The results obtained in this experiment showed that both of the representations performed very similar to each other with neither of scoring well below 0.25 as measured by both precision and F1 across the value of K and number of recommendations. However, this experiment provided the research with a benchmark to work with. Experiment 2 proved critical as it showed that the number of words used to develop embedding is important as insufficiency of word examples to learn from, despite having high vocabulary intersection could provide poor performance. Domain-specific embedding had lesser word examples while having high vocabulary intersection but higher word example and slightly lower vocabulary intersection with pre-trained embedding provided significantly higher performance. The highest result was achieved using pre-trained embedding with resultant vectors with precision around 0.5 and recall around 0.6 with F1 aggregating these around 0.54. Progressing forward with FastText another word embedding algorithm with domain-

specific data to develop word embedding in experiment 3 provided no worth in comparison to what was achieved by pre-trained embedding in experiment 2.

Experiment 4 was a slight turn around for domain-specific embedding when WMD was used instead of cosine similarity. The results appeared promising with 0.28 - precision, 0.44 - recall and F1 aggregating these scores to around 0.34. However, in this experiment, only a sample of 26 judgements was used due to computational complexities associated with WMD. Later, experiment 5 concluded by comparing WMD and cosine based approaches with best of both being FastText CBoW and word2vec pre-trained respectively. The results of this experiment showed that WMD being a distance measure has an advantage with less nearest neighbours and cosine has an advantage with more nearest neighbours as observed throughout the research. A detailed investigation must be done to conclude which of the approach works well.

Chapter 5

Conclusion

This chapter provides a brief overview of the research problem followed by the design and experimental results obtained with key findings. Finally, the chapter looks at the contribution of this research and offers some potential directions that could lead to further investigation towards solving the research problem.

5.1 Overview of work

The research started with an aim to investigate traditional and state-of-the-art techniques in an attempt to solve the real-world problem of recommending labels for Irish legal judgements. Two independent datasets were used in this research. Labelled Irish judgements were provided by *Stare Decisis* and the other domain-specific text data was scraped from the internet from two sources. Basic pre-processing steps were applied on both the datasets appropriately before proceeding to implementation.

This research was empirical in nature and sought knowledge that is necessary to solve the problem in the fixed narrow band of techniques chosen. Five individual experiments were designed to investigate and understand the feasibility of the techniques in focus. The first experiment was devised to compare traditional text representations while setting a baseline for other techniques yet to be implemented. The second experiment focused completely on the word2vec algorithm by utilizing an already available

pre-trained embedding and also by developing one using domain-specific text data. The third experiment was designed to focus on FastText algorithm to develop word embedding for an advantage that it works on character grams rather the words themselves which are beneficial when working with less frequent words. The fourth utilized word mover's distance instead of cosine which was used until now in the research. Both the word2vec and FastText were used in conjunction with WMD in this experiment. The final experiment was devised to verify if the best of cosine based approach works better than WMD based approach.

One of the challenges encountered during this research was the amount of time that was required to build the recommendation systems. More than 500 recommendation systems were built across all experiments consuming over 300 man-hours of development time with multiple python environments equipped with Graphical Processing Units (GPU).

Another challenge faced was with the development of the recommendation system using WMD. One of the research objectives was to compare WMD based recommendation system to cosine based recommendation system, however, in practice, this proved infeasible to reliably compare both the approaches due to the amount of computation that is required. The major issue was that the sample chosen was too small to accept the results confidently.

The dataset initially received from *Stare Decisis* was plagued with infrequent words. It was found that 30% of the words in the corpus had a frequency of just 1. Another challenge with the dataset is about the uncertainty of the presence of bias, it can be inferred from what is visible in the charts however, this can easily complicate the understandings of the results. For this research, it is ignored and all the recommendation systems developed were evaluated on the same label frequency distribution as received.

Due to computational complexities associated with the word embeddings, hyperparameter space was not searched for better parameters. All of the algorithms were set to default as provided by Gensim (Řehůřek & Sojka, 2010) which is also similar to

the parameters of pre-trained embedding.

5.2 Research findings

Feature representation and its quality play a key role in machine learning and the same was proven in this research. The performance achieved using traditional representations were poor and failed to set a good benchmark for models that were created later.

Domain-specific embeddings despite having the maximum of 66% vocabulary intersection struggled with cosine similarity approach as the quality of word vectors were poor due to less amount of training examples for the model to learn from. Pre-trained embedding, however, performed significantly better than domain-specific embeddings even with lesser vocabulary intersection in comparison due to the quality of word vector representation that was trained on 100 Billion words.

WMD has shown glimpses of what it could be capable of to solve this problem. It emerged as a better performer on a sample data with domain-specific embedding even against the pre-trained embeddings with proved to contain quality word vectors. It will be interesting to investigate more on this in the legal domain.

5.3 Impact of the research

As clearly stated by (Wolpert & Macready, 1997) there is no free lunch in machine learning and only through a thorough exploration of numerous data pre-processing, feature representation, and modelling techniques can an effective approach be achieved. With every step of development in the field of machine learning being vast on their own, the problem identified must be attempted to solve from somewhere.

The problem of labelling Irish legal documents can be solved in multiple ways and this research attempted to do this by employing traditional and current state-of-the-art approaches. This research can be considered as an initial exploration done for the research problem.

TF and TF-IDF vectors take longer time to generate recommendations and are also inefficient on this dataset.

Word embeddings require a massive amount of data to learn a meaningful representation and was proved by experiment 2. Domain-specific embeddings could achieve better performance with WMD as compared to cosine similarity in KNN based recommendation system.

Developing Irish legal domain-specific word embeddings to solve this research problem could be challenging as the amount of available open legal text could prove insufficient.

Application of KNN recommendation system approach is feasible for this dataset when cosine similarity is used both with TF and TF-IDF vectors and also word embeddings.

Exploration has been done through an evidence-based approach for the sake of gathering knowledge that could lead to the identification of an efficient approach for the problem.

5.4 Future work and recommendations

Number of directions are available to extend this work.

- One obvious option is to gather more data to develop and improve the quality of word embeddings.
- Employing other metrics such as Relaxed Word Movers Distance(RWMD) and Word Centroid Distance (WCD) which are very similar to WMD will be interesting as these theoretically require less computation in comparison to WMD to calculate the distance. These measures are yet to be launched by Gensim.
- GloVe is another interesting algorithm by Stanford and this can also be explored to find its suitability for the problem.
- The amount of computation required to develop word embedding and test it in an application is truly challenging and research with focus only on arriving at the best parameters can potentially step closer to solving this problem.

- Essential component to develop a word embedding is the amount of text data. For legal domain embedding in English, gathering more data from all of the open legal websites who maintain their records in English could significantly improve the quality of word vectors.
- Re-labelling all of the judgments received by several solicitors and aggregating can reduce the bias and improve the quality of the labels assigned.
- Sampling techniques can also be used to tackle these kinds of situation where class labels are imbalanced.
- Several variants of TF-IDF have been developed and available and further investigation can also be done using these.
- Dimensional reduction techniques can be employed on TF-IDF vectors to reduce sparsity and explore the possibilities on this dataset.
- Memory based recommendation system was developed in this research and model-based approaches are also a feasible option.
- Word sense disambiguation is an interesting option to incorporate with word embeddings to help find difference between words such as interest, apple and lotus that when written are the same but mean differentl in different contexts.

Bibliography

Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing Management*, *39*(1), 45 - 65. Retrieved from http://www.sciencedirect.com/science/article/pii/S0306457302000213 doi: https://doi.org/10.1016/S0306-4573(02)00021-3

Bastani, K., Namavari, H., & Shaffer, J. (2019). Latent Dirichlet allocation (LDA) for topic modeling of the CFPB consumer complaints. *Expert Systems with Applications*, *127*, 256 – 271. Retrieved from http://www.sciencedirect.com/science/article/pii/S095741741930154X doi: https://doi.org/10.1016/j.eswa.2019.03.001

Bengio, Y., Ducharme, R., & Vincent, P. (2001). A Neural Probabilistic Language Model. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in Neural Information Processing Systems 13* (pp. 932–938). MIT Press. Retrieved 2019-12-25, from http://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003, March). A neural probabilistic language model. *J. Mach. Learn. Res.*, *3*(null), 1137–1155.

Bhardwaj, A., Di, W., & Wei, J. (2018). *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd. (Google-Books-ID: ISBKDwAAQBAJ)

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender sys-

tems survey. *Knowledge-Based Systems*, *46*, 109 – 132. Retrieved from http://www.sciencedirect.com/science/article/pii/S0950705113001044 doi: https://doi.org/10.1016/j.knosys.2013.03.012

Cancho, R. F. i., & Solé, R. V. (2003). Least effort and the origins of scaling in human language. *Proceedings of the National Academy of Sciences*, *100*(3), 788–791. Retrieved from https://www.pnas.org/content/100/3/788 doi: 10.1073/pnas.0335980100

Chapter 9 - Functional Networks. (2005). In E. Castillo, A. Iglesias, & R. Ruíz-Cobo (Eds.), *Functional Equations in Applied Sciences* (Vol. 199, pp. 169 – 232). Elsevier. Retrieved from http://www.sciencedirect.com/science/article/pii/S0076539205800128 doi: 10.1016/S0076-5392(05)80012-8

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms*. MIT Press.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern Classification*. John Wiley & Sons. (Google-Books-ID: Br33IRC3PkQC)

Eberhart, R. C., & Shi, Y. (2007). chapter six - Neural network implementations. In R. C. Eberhart & Y. Shi (Eds.), *Computational Intelligence* (pp. 197 – 267). Burlington: Morgan Kaufmann. Retrieved from http://www.sciencedirect.com/science/article/pii/B9781558607590500068 doi: 10.1016/B978-155860759-0/50006-8

Fathi, E., & Shoja, B. M. (2018). Chapter 9 - Deep Neural Networks for Natural Language Processing. In V. N. Gudivada & C. R. Rao (Eds.), *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications* (Vol. 38, pp. 229 – 316). Elsevier. Retrieved from http://www.sciencedirect.com/science/article/pii/S016971611830021X doi: 10.1016/bs.host.2018.07.006

Friedman, N., Geiger, D., & Goldszmidt, M. (1997, November). Bayesian Network Classifiers. *Machine Learning*, *29*(2), 131–163. Retrieved from https://doi.org/10.1023/A:1007465528199 doi: 10.1023/A:1007465528199

Gronvall, P., Huber-Fliflet, N., Zhang, D. J., Keeling, R., Neary, R., & Zhao, D. H. (2018, December). An Empirical Study of the Application of Machine Learning and Keyword Terms Methodologies to Privilege-Document Review Projects in Legal Matters. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 3282–3291). (ISSN: null) doi: 10.1109/BigData.2018.8621945

Huang, G., Guo, C., Kusner, M. J., Sun, Y., Sha, F., & Weinberger, K. Q. (2016). Supervised Word Mover\textquotesingle s Distance. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29* (pp. 4862–4870). Curran Associates, Inc. Retrieved from `http://papers.nips.cc/paper/6139-supervised-word-movers-distance.pdf`

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015, November). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. Retrieved 2019-12-24, from `http://www.sciencedirect.com/science/article/pii/S1110866515000341` doi: 10.1016/j.eij.2015.06.005

Kotu, V., & Deshpande, B. (2015). Chapter 9 - Text Mining. In V. Kotu & B. Deshpande (Eds.), *Predictive Analytics and Data Mining* (pp. 275 – 303). Boston: Morgan Kaufmann. Retrieved from `http://www.sciencedirect.com/science/article/pii/B9780128014608000094` doi: 10.1016/B978-0-12-801460-8.00009-4

Kotu, V., & Deshpande, B. (2019a, January). Chapter 4 - Classification. In V. Kotu & B. Deshpande (Eds.), *Data Science (Second Edition)* (pp. 65–163). Morgan Kaufmann. Retrieved 2019-12-28, from `http://www.sciencedirect.com/science/article/pii/B9780128147610000046` doi: 10.1016/B978-0-12-814761-0.00004-6

Kotu, V., & Deshpande, B. (2019b). Chapter 7 - Clustering. In V. Kotu & B. Deshpande (Eds.), *Data Science (Second Edition)* (Second Edition ed., pp. 221 – 261). Morgan Kaufmann. Retrieved from `http://www.sciencedirect.com/science/article/pii/B9780128147610000071` doi: 10.1016/B978-0-12-814761-0.00007-1

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015, 01). From word embeddings to document distances. *Proceedings of the 32nd International Conference on Machine*

Learning (ICML 2015), 957-966.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on international conference on machine learning - volume 32* (p. II–1188–II–1196). JMLR.org.

Leonard, D. (2019). *Machine learning spec9270 - week1.* University Lecture.

Li, B., & Han, L. (2013). Distance Weighted Cosine Similarity Measure for Text Classification. In H. Yin et al. (Eds.), *Intelligent Data Engineering and Automated Learning – IDEAL 2013* (pp. 611–618). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-41278-3_74

Li, Z. (2019, January). A Classification Retrieval Approach for English Legal Texts. In *2019 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS)* (pp. 220–223). (ISSN: null) doi: 10.1109/ICITBS.2019.00059

LUNTZ, A. (1969). On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*. Retrieved from `https://ci.nii.ac.jp/naid/10015536586/en/`

Metcalf, L., & Casey, W. (2016, January). Chapter 2 - Metrics, similarity, and sets. In L. Metcalf & W. Casey (Eds.), *Cybersecurity and Applied Mathematics* (pp. 3–22). Boston: Syngress. Retrieved 2019-12-28, from `http://www.sciencedirect.com/science/article/pii/B9780128044520000026` doi: 10.1016/B978-0-12-804452-0.00002-6

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.* Retrieved from `http://arxiv.org/abs/1301.3781`

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems -*

Volume 2 (pp. 3111–3119). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2999792.2999959 (event-place: Lake Tahoe, Nevada)

Mitchell, T. M. (1997). *Machine Learning* (1st ed.). New York, NY, USA: McGraw-Hill, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical methods in natural language processing (emnlp)* (pp. 1532–1543). Retrieved from http://www.aclweb.org/anthology/D14-1162

Powers, D. M. W. (1998, November). Applications and explanations of Zipf's law. In (pp. 151–160). Association for Computational Linguistics. Retrieved 2019-12-21, from http://dl.acm.org/citation.cfm?id=1603899.1603924

Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., & Riedl, J. (2002). Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (pp. 127–134). New York, NY, USA: ACM. Retrieved 2019-12-24, from http://doi.acm.org/10.1145/502716.502737 (event-place: San Francisco, California, USA) doi: 10.1145/502716.502737

Řehůřek, R., & Sojka, P. (2010, May 22). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta: ELRA. (http://is.muni.cz/publication/884893/en)

Ross, S. M. (2017). Chapter 12 - Linear Regression. In S. M. Ross (Ed.), *Introductory Statistics (Fourth Edition)* (Fourth Edition ed., pp. 519 – 584). Oxford: Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128043172000126 doi: 10.1016/B978-0-12-804317-2.00012-6

Samuel, A. L. (1959, July). Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, *3*(3), 210–229. Retrieved from https://doi.org/10.1147/rd.33.0210 doi: 10.1147/rd.33.0210

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). The Adaptive Web. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), (pp. 291–324). Berlin, Heidelberg: Springer-Verlag. Retrieved 2019-12-24, from http://dl.acm.org/citation.cfm?id=1768197.1768208

Shi, G. (2014, January). Chapter 5 - Decision Trees. In G. Shi (Ed.), *Data Mining and Knowledge Discovery for Geoscientists* (pp. 111–138). Oxford: Elsevier. Retrieved 2019-12-26, from http://www.sciencedirect.com/science/article/pii/B9780124104372000059 doi: 10.1016/B978-0-12-410437-2.00005-9

Tao, Y., Cui, Z., & Wenjun, Z. (2018, Oct). A multi-label text classification method based on labels vector fusion. In *2018 international conference on promising electronic technologies (icpet)* (p. 80-85). doi: 10.1109/ICPET.2018.00021

Theodoridis, S., & Koutroumbas, K. (2009). Chapter 13 - Clustering Algorithms II: Hierarchical Algorithms. In S. Theodoridis & K. Koutroumbas (Eds.), *Pattern Recognition (Fourth Edition)* (Fourth Edition ed., pp. 653 – 700). Boston: Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9781597492720500153 doi: 10.1016/B978-1-59749-272-0.50015-3

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., & Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, *73*, 185–214. doi: 10.1007/s10994-008-5077-3

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). Chapter 4 - Algorithms: The basic methods. In I. H. Witten, E. Frank, M. A. Hall, & C. J. Pal (Eds.), *Data Mining (Fourth Edition)* (Fourth Edition ed., pp. 91 – 160). Morgan Kaufmann. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128042915000040 doi: 10.1016/B978-0-12-804291-5.00004-0

Wolpert, D. H., & Macready, W. G. (1997, April). No free lunch theorems for optimization. *Trans. Evol. Comp*, *1*(1), 67–82. Retrieved from https://doi.org/10.1109/4235.585893 doi: 10.1109/4235.585893

Yahyaoui's, A., Yahyaoui, I., & Yumuşak, N. (2018a, January). 13 - Machine Learning Techniques for Data Classification. In I. Yahyaoui (Ed.), *Advances in Renewable Energies and Power Technologies* (pp. 441–450). Elsevier. Retrieved 2019-12-26, from http://www.sciencedirect.com/science/article/pii/B9780128131855000097 doi: 10.1016/B978-0-12-813185-5.00009-7

Yahyaoui's, A., Yahyaoui, I., & Yumuşak, N. (2018b, January). 13 - Machine Learning Techniques for Data Classification. In I. Yahyaoui (Ed.), *Advances in Renewable Energies and Power Technologies* (pp. 441–450). Elsevier. Retrieved 2019-12-26, from http://www.sciencedirect.com/science/article/pii/B9780128131855000097 doi: 10.1016/B978-0-12-813185-5.00009-7

Yang, X.-S. (2019a). 5 - Logistic regression, PCA, LDA, and ICA. In X.-S. Yang (Ed.), *Introduction to Algorithms for Data Mining and Machine Learning* (pp. 91 – 108). Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128172162000120 doi: 10.1016/B978-0-12-817216-2.00012-0

Yang, X.-S. (2019b). 6 - Data mining techniques. In X.-S. Yang (Ed.), *Introduction to Algorithms for Data Mining and Machine Learning* (pp. 109 – 128). Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128172162000132 doi: 10.1016/B978-0-12-817216-2.00013-2

Yang, X.-S. (2019c). 6 - Data mining techniques. In X.-S. Yang (Ed.), *Introduction to Algorithms for Data Mining and Machine Learning* (pp. 109 – 128). Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128172162000132 doi: 10.1016/B978-0-12-817216-2.00013-2

Yang, X.-S. (2019d). 7 - Support vector machine and regression. In X.-S. Yang (Ed.), *Introduction to Algorithms for Data Mining and Machine Learning* (pp. 129 –

138). Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128172162000144 doi: 10.1016/B978-0-12-817216-2.00014-4

Yu, B., Xu, Z.-b., & Li, C.-h. (2008). Latent semantic analysis for text categorization using neural network. *Knowledge-Based Systems*, *21*(8), 900 – 904. Retrieved from http://www.sciencedirect.com/science/article/pii/S0950705108000993 doi: https://doi.org/10.1016/j.knosys.2008.03.045

Zhang, H., Kiranyaz, S., & Gabbouj, M. (2017, March). A k-nearest neighbor multilabel ranking algorithm with application to content-based image retrieval. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2587–2591). (ISSN: 2379-190X) doi: 10.1109/ICASSP.2017.7952624

Appendix A

Hyper-parameters

A.1 Word2vec/FastText

- "window": ["300"]
- "min_count":[2]
- "size":[50]
- "negative":[5]
- "alpha":[0.05]
- "min_alpha":[0.0001]

Appendix B

Dataset

Labelled judgements : Provided by *Stare Decisis*

scraped dataset:

Source1: `www.courts.ie`

Source2: `www.supremecourt.gov/oral_arguments/argument_transcript/2019`