UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Master Degree in Computer Science

Mining Terminologies in Text

ADVISOR:
Alfio Ferrara

CANDIDATE:
Colin Fernandes
914777

Academic Year 2019/2020

You shall know a word by the company it keeps.
- John Rupert Firth [Firth, 1957]

Abstract

Terminologies are essential to identify, update, organise information entities. Identifying these terminologies form records, documents, or any other information source is key to improving the quality of the information entity. Ontology is an example of one such Information entity. It is an efficient and intuitive way of representing a domain. There is a ton of literature to identify terminologies from texts, but there is hardly any literature to discern semantic and contextual relations between them. Currently, the task of finding these related words is user-dependent; hence it is inconsistent, time-consuming and expensive.

This thesis proposes a novel method to mine related terminologies from given a corpus. First, the comprehensive overview of techniques in Automatic Term Recognition using Linguistic and statistical methods are summarised. Adding to the summary, state of the art on text representation and measures for text similarity are reported. Next, the terminologies are mined based on the existing literature, and a novel method is proposed and implemented to extract terminologies based on semantic relatedness to a query. The method is evaluated against data-sets with existing ground truth datasets and on Illinois Case Law opinions. Findings of interest are reported in the results.

Contents

Abstract

1 Introduction **1**
1.1 Problem Statement . . . 2
1.2 Research Objective . . . 4
1.3 Outline of the Thesis . . . 4

2 State Of The Art **7**
2.1 Automatic Terminology Recognition (ATR) . . . 7
2.1.1 Terminology Features . . . 8
2.1.1.1 Syntactic Features . . . 8
2.1.1.2 Semantic Features . . . 8
2.1.1.3 Pragmatic features . . . 8
2.1.2 Linguistic Filtering . . . 9
2.1.3 Statistical Filtering . . . 9
2.1.3.1 TF-IDF . . . 9
2.1.3.2 Residual-IDF (R-IDF) . . . 11
2.1.3.3 C-Value\NC-Value . . . 11
2.1.3.4 Basic . . . 12
2.1.3.5 ComboBasic . . . 12
2.1.4 Semantic Relatedness between Terms . . . 13
2.2 Text Representations . . . 13
2.2.1 Neural Probabilistic Language Model . . . 13
2.2.2 Word2Vec . . . 13
2.2.3 Global Vectors (GloVe) . . . 14
2.2.4 Non-Parametric Multi-Sense Skip Gram(NP-MSSG) . . . 14
2.2.5 FastText . . . 15
2.2.6 ELMo . . . 15
2.2.7 BERT . . . 15
2.3 Similarity Measures . . . 18
2.3.1 Character-Based Similarity Measures . . . 18
2.3.2 Term-Based Similarity Measures . . . 18
2.3.2.1 Manhattan Distance . . . 18
2.3.2.2 Dice's Coefficient . . . 18
2.3.2.3 Euclidean Distance . . . 18

2.3.2.4 Jaccard Similarity . . . 19
2.3.2.5 Word Movers Distance . . . 19
2.3.2.6 Cosine Similarity . . . 19

3 Methodology **21**
3.1 Methodology Overview . . . 21
3.2 Automatic Terminology Recognition . . . 21
3.2.1 Data Collection . . . 21
3.2.2 Data Tokenization . . . 21
3.2.3 Data Preprocessing . . . 22
3.2.4 Parts of Speech Identification . . . 22
3.2.5 Linguistic Filtering . . . 23
3.2.5.1 POS Patterns for Candidate Extraction . . . 24
3.2.6 Statistical Filtering . . . 25
3.2.6.1 TF-IDF . . . 26
3.2.6.2 Residual IDF . . . 26
3.2.6.3 C-Value/NC- Value . . . 26
3.2.6.4 Basic/ComboBasic . . . 26
3.3 Semantically Relatedness of Terms . . . 29
3.3.1 Word Embedding Training . . . 29
3.3.2 Term Embeddings . . . 30
3.3.3 Discerning Similarity Information . . . 31
3.3.4 Context Information Mining . . . 32

4 Experimentation and Results **35**
4.1 Pre-Text . . . 35
4.2 Implementation Tools . . . 35
4.3 Comparison with existing Ground Truth . . . 35
4.3.1 Linguistic Filtering Pattern Analysis . . . 36
4.3.2 Statistical Filtering . . . 37
4.3.3 Evaluation Metrics . . . 37
4.3.4 Conclusion from Testing over Term annotated datasets . . . 42
4.4 Evaluation of Embedding Models and Similarity Measures . . . 43
4.4.1 Sample Text Corpus . . . 43
4.4.1.1 The 20 Newsgroups Data-set . . . 43
4.4.1.2 Leipzig Wikipedia Corpora . . . 44
4.4.2 Selection of Embedding Algorithm . . . 44
4.4.3 Conclusion on Evaluation of Embedding Models with Cosine Similarity . . . 44

5 Implementation and Issues **47**
5.1 Implementation on Illinois Dataset . . . 47
5.1.1 Data Collection . . . 47
5.1.1.1 Memory Issue . . . 48
5.1.2 Tokenizing text . . . 48

5.1.2.1 Tokenizer Issues . . . 49
5.1.3 Parts of Speech Tagging . . . 49
5.1.3.1 Tagging Issues . . . 49
5.1.4 Linguistic Filtering using Noun Phrase Chunking . . . 51
5.1.4.1 Chunk Grammar Pattern Analysis on Illinois Dataset . . . 51
5.1.4.2 Linguistic Filtering Issues . . . 55
5.1.5 Statistical Filtering of Candidate Term . . . 56
5.1.5.1 Issues with Statistical Measures . . . 56
5.2 Semantic Information Extraction . . . 56
5.2.1 Feature Extraction of Terms . . . 56
5.2.1.1 Word2Vector Training using FastText . . . 62
5.2.1.2 Word2Vector Using Pre-Trained Wikipedia Embeddings . 62
5.2.1.3 Issues with Featurizing Terms . . . 62
5.2.2 Similarity Querying . . . 62
5.2.3 Contextual Querying . . . 63
5.2.3.1 WordNet . . . 63
5.2.4 Issues with Semantic Information Extraction . . . 65

6 Conclusion **69**
6.1 Future Implementation . . . 69

Bibliography **75**

Appendix A **77**

Appendix B **83**

Abbreviations **85**

List of Algorithms **87**

List of Figures **89**

List of Tables **91**

Acknowledgements **93**

Chapter 1

Introduction

"Terminology" is a system of nomenclature belonging to a particular field or domain. It can be specific to an area of study, a system or a profession. These terminologies are technical words or jargon's used to aid swift comprehension of tasks and labels specific to respective domains. For instance, words like 'nouns', 'verbs', 'articles', 'synonyms', 'hyponyms' are all terms which are related to English grammar. Nouns, Verbs make the 'parts of speech', whereas 'synonyms', 'hyponyms' describes the relationship between terms. Terms are classified into two types [Wong, 2009]. Namely, simple terms (i.e. single-word terms) and complex terms (multi-word terms), and collectively, both types of terms constitute what is known as Terminology.

Terminologies have higher possibilities of distinguishing similar concepts with different meanings, as they can be grammatically perceived as noun phrases in a sentence. Terms are also useful in avoiding misinterpretation of the meaning, for instance, when two different names refer to the same concept, or a single name represents two different concepts. A use case of Terminologies can be exemplified in the domain of legal language. Precise terminology is the hallmark of successful disambiguation or avoiding misinterpretation of judgements. Such a unique property can be exploited to augment an information retrieval logic on existing Case Law[1] dataset. Such an area of study is called 'Legal information retrieval' [2]. An effective system to mine intended legal information would revolutionize the legal fraternity to systematise and efficiently track precedents for ongoing legal conflicts, or even for a common person who can lookup a similar case type and decide their future course of action.

Terms and the tasks related to their recognition are an integral part of many applications that deal with natural language tasks such as large-scale search engines, automatic thesaurus construction, machine translation and ontology learning for purposes ranging from indexing to cluster analysis[Wong, 2009]. More specific and purposeful interpreta-

[1] 'Case law' or 'Precedent' or 'Stare decisis'—a Latin phrase meaning "let the decision stand", is the collection of past legal decisions written by courts and similar tribunals in the course of deciding cases, in which the law was analysed using these cases to resolve ambiguities for deciding current cases

[2] Legal information retrieval is the science of information retrieval applied to legal text, including legislation, case law, and scholarly works[Maxwell and Schafer, 2008] Accurate legal information retrieval is important to provide access to the law to laymen and legal professionals. Its importance has increased because of the vast and quickly increasing amount of legal documents available through electronic means[Jackson et al., 1998]. Legal information retrieval is a part of the growing field of legal informatics.

tions of terms do exist for specific applications such as ontology learning. Ontologies are an efficient way of representing the knowledge of a domain. It is a data model that represents a set of concepts within a domain and the relationships between those concepts. WordNet[Miller, 1995] is an excellent example of an ontology. It is a lexical database having terms exhibiting relations with other terms or having properties, as mentioned in the previous example. Systems based on knowledge resources like ontologies have 'cognitive architecture' meaning, there is a decision architecture that natively exploits knowledge representation and intelligent inference based on it. Such knowledge resources give power to model, reason and manage complex data systems from different domains. A famous example is IBM's question answering system, 'Watson' which uses the knowledge-base 'DBPedia' to augment its question answering system. As in an earlier example of Case-Law ontology could be designed to graphically relate knowledge across the legal spectrum. To build such a knowledge resource, finding related-words from a corpus is the primary task.

Terminologies can be easily misunderstood for Keywords[Scott and Tribble, 2006]. 'Keyword' is a word which occurs in a text more often than we would expect to occur by chance alone. Keywords are generated by carrying out a statistical test (e.g., log-linear or chi-squared) which compares the word frequencies in a text against their expected frequencies derived in a much larger corpus [3]. This acts as a reference for general language use. Terms[4] can be a part of the keywords, but it is not the same the other way around. 'Keyness' is a textual feature, whereas 'Termness' is an information feature.

1.1 Problem Statement

One of the first steps to model a knowledge domain is to collect a vocabulary of domain-relevant terms, constituting the linguistic surface manifestation of domain concepts. Knowledge perceptive systems are generally designed either using it in combination with an external knowledge source (like in case of 'IBM - Watson with 'DBPedia') or by statistically inferring the current data. For building a knowledge source Term recognition is one of the primary tasks. As mentioned earlier, recognizing terminologies from documents is knowledge dependent task. Knowledge perception is mostly a human trait, and up until now, there is very little progress on improving the accuracy of knowledge perceptive machines. Processes like these are manually performed. A process which has manual intervention leads to inconsistencies as different people represent these terminologies in their own specific and unique ways. These often need fine-tuning as they become incompatible within the same project. Such a process is inefficient, time-consuming, expensive and often presents a challenge to a domain taxonomist.

The task of recognizing terms through textual data crunching is called Term mining or Automatic Term Extraction(ATE) or Automatic Term Recognition(ATR) [5]. The research

[3] `https://en.wikipedia.org/wiki/Keyword_(linguistics)`

[4] Terms and Terminologies are used interchangeably in this litrature

[5] An area related to the study, design and development of techniques for the extraction of stable lexical units from the text, and the filtering of these lexical units, usually through some scoring and ranking schemes, for the identification of terms. Not considering the possibility of minor differences imposed by different researchers, this area may also be referred to as term extraction or terminology mining

in terms, and their recognition can be traced back to the field of Information Retrieval (IR)[Kageura and Umino, 1996]. Although much work has been done on ATR, the state of the art performance is still low, as mentioned in the literature by [Hasan and Ng, 2014]. Other significant factors that affect ATE performance measures are the length of documents, a lack of structural consistency (for instance, a lack of a defined index or abstract), topic changes, and the presence of uncorrelated topics in the same text. The main aim in term recognition is to determine whether a word or phrase is a term which characterises the target domain. This fundamental question can be further decomposed to reveal two critical notions in this area, namely, unithood[6] and termhood[7][Wong, 2009].

Linguistic knowledge in Natural Language understanding systems is commonly stratified across several levels. This is true of Information Extraction as well. Typical state-of-the-art Information Extraction systems require syntactic-semantic patterns for locating facts or events in the text; domain-specific word or concept classes for semantic generalization; and a specialized lexicon of terms that may not be found in general-purpose dictionaries, among other kinds of knowledge. Hence adding to the above problem of ATR in the previous paragraph, there is also a need to group terms that exhibit properties of semantic relatedness. In this literature, these are denoted as related terms. Multiple terms can relate at a certain level in knowledge structure. This can be identically explained as siblings at a level in the graph structure of ontological learning. Finding semantic relations within text or words is a prevalent task, and a ton of literature can be found in this regard. The recent and comparably efficient models can be found in the literature on word embeddings, Where relations between words can be easily discerned as they are transformed into numerical vectors. Words as vectors can be mathematically perceived as points on the n-dimensional vector space, and relations can be interpreted as the distance between the points in the same vector space. Although finding semantic relations for single-word terms is not challenging. It does create a challenge for terminologies as they contain multi-word terms since, in the case of Terms, similarities between multi-word terms also need to be discerned.

The embedding algorithm can be trained for (1 to n)-grams, but this reduces the efficiency of the embedding algorithm considerably as it has to train vectors for $V*(\frac{n(n-1)}{2})$ words [8], where V is the vocabulary of the dataset used to train word embeddings and 'n' is the number of words in the terms. The number of words in term can vary from 1 to 6. The domain of Arts, for example, terms tend to be shorter than in science and technology. Terms that only consist of nouns, for example, very rarely contain more than 5 or 6 words.[Frantzi et al., 2000]. Similarly, as in the previous paragraph, it is also crucial to identify super-subordinate and lower-subordinate levels that can aid in tree-based information among terms.

[Wong, 2009]

[6]Unithood concerns with whether or not sequences of words should be combined to form more stable lexical units

[7]Termhood is the extent to which these stable lexical units are relevant to some domains

[8]Calculated by the arithmetic mean $\frac{n(n-1)}{2}$ Eg: A term of 3 words "Principle Component Analysis" will have 'Principle', 'Component', 'Analysis', 'Principle Component', 'Component Analysis' and 'Principle Component Analysis'. Total of 6 terms for a 3-gram term

1.2 Research Objective

With the open problems presented at the end of the previous section, this thesis aims to explore ways by using natural language processing techniques to find an effective method to generate candidate terms and discern semantic and contextual relations between them.

The central hypothesis of the thesis is that the Automatic Term Recognition can be successfully performed in any domain by extracting single and multi-word terms through linguistic filtering of noun - phrases and then applying statistical filter over the candidate terms to filter out probable non - terms. Post extraction of terms the relations between them are discerned by calculating the distance between terms in a vector space. Finally, contextual or subject information is also mined using a pre-existing knowledge source in the same domain or a generic domain.

The research objective is achieved by evaluating the ATR results datasets with existing ground truth and later implemented on the Case Law dataset. The process of the ATR is as follows. First, a linguistic filter is built by using patterns of Parts of Speech tags, which extracts the candidate terms in the corpus. Next, the candidates are filtered statistically by one of these measures {TF-IDF, C- Value\NC- Value, R-IDF, Combo and Basic}. Finally, the relations between terms are recognized by finding cosine distance between the word transformed vectors.

The novelty of the thesis lies in the extraction of related terms and identifying various levels of the relation between them. The whole process is implemented on an Illinois Case Law bulk dataset, the terms are extracted, and later the relation between the terms are identified using fastText embedding algorithm. Contextual relations identified using semantically identical relations from WordNet knowledge base. Finally, observations of interest are reported.

1.3 Outline of the Thesis

Chapter 2 Explains the current state of the art and how it fits into the master thesis. Starts from the various methods in ATR using linguistic and statistical methods, followed by surveying on embedding models. The pros and cons of all the models are compared and tabulated.

Parallelly covers the existing literature on text similarity measures of which divided into three categories, based on the way similarity is calculated. Brief description of each of the measures are mentioned.

Chapter 3 Describes the methodology and steps involved in the proposed term mining hypothesis. Explains the architecture of the ATR Model and Model to identify relations. Post recognition of term ways to recognize term relatedness and context relatedness.

Chapter 4 Describes the Experimentation and evaluation of the hypothesis. The ATR and embedding algorithms are evaluated by verifying against datasets with existing ground truth. The results of the ground truth evaluations are reported here.

Adding to the prior, embedding models are explored on two of the publicly available datasets. Analysis of results from the tests is reported. Finally, concluded by selecting a candidate embedding model for the term relatedness implementation.

Chapter 5 Implementation of the Hypothesis on Illinois dataset and tabulation of interesting observations.

Chapter 6 Concludes by analysing the work during the process of achieving the described goals and proposes additional work which the thesis has opened up for the future.

Chapter 2

State Of The Art

2.1 Automatic Terminology Recognition (ATR)

Automatic Terminology Recognition(or Extraction ATE) deals with the extraction of terms, words and collocations representing domain-specific concepts - from a corpus of unstructured domain-specific text. It is a fundamental task for knowledge mining, often a preprocessing step to many complex Natural Language Processing tasks. These can include information retrieval, cold start knowledge base population, glossary construction, ontology creation, text summarization, machine translation, topic detection, and ultimately enabling business intelligence.

ATR is still an unsolved problem [Astrakhantsev, 2018] and a ton of methods have been developed over the years to cope with increasing demand for automated sense-making of the ever-growing number of specialized documents in scientific, industrial, governmental and digital archives. Some of the literature in the timeline of ATR are described in Table 2.1. Most of the methods are built by using two main characteristics of terminologies. First, by extraction of candidate terms using linguistic filters(nouns, noun phrase, n-grams). Second, by eliminating terms which show less probability to be a term based on scoring using statistical measures(TF-IDF).

Despite a ton of research in the area and plethora of methods introduced, limitations in these persist. As per [Zhang et al., 2008], these limitations are mainly categorised into two types; First, no method performs consistently in all situations. Second, while state-of-the-art typically makes use of features such as word statistics (e.g., frequency) to score candidate terms, they often overlook the role of semantic relatedness. For example, 'cat' is similar to 'dog', and is related but not similar to fur. To illustrate the usefulness of semantic relatedness in the context of ATR, assuming protein a representative term in a biomedical corpus, then the scores of words highly related to it such as polymer and nitrogenous should be boosted according to their degree of relatedness with protein, in addition to their frequency[Zhang et al., 2018]. In the following section, a methodology is described to gather terms of similar semantic relations.

2.1.1 Terminology Features

Terminology is defined by describing its features. As a consequence, it supports in distinguishing the term from ordinary words. Such features can serve as a basis for methods of Automatic Term recognition. Up until now, there are a significant number of ways formulated to classify features of terms. In [Astrakhantsev et al., 2015], the literature classifies terms into three aspects. All these features combined make a clear description of a term.

2.1.1.1 Syntactic Features

It describes the morphological aspects of the terms. This feature is given by four subclasses.

- Nominativeness: nouns or noun-based word combinations are generally regarded as terms.
- Normativeness: conformity with language norms.
- Terminological invariance: The absence of diversity in writing and pronouncing the term to avoid hindrance in communication between specialists.
- Motivationess, or self-explainability: Maximum relation between domain concept and the term

2.1.1.2 Semantic Features

It describes the intention of the term.

- Systemacy: the term belongs to certain terminology, to a system of concepts of a specific domain or field of knowledge
- Correspondence to the concept denominated: the absence of contradictions between the lexical meaning of the words constituting the term and the meaning of the term in a given terminology.
- Unambiguity: monosemy, of the term: uniqueness of the term in a given terminology
- Intensional exactness: exactness and boundedness of the term meaning

2.1.1.3 Pragmatic features

Describes the specificity of the term behaviour:

- Introducability: not requiring a description to describe the term.
- Definiteness since the intensional exactness of the term (see above) is generally achieved by finding a scientific definition, the definition itself can serve as a feature of the term.

- Independence from the context: this feature follows from monosemy of the term; it can be said that the terminology to which the term belongs serves as a context of the term, which defines its meaning.
- Variational stability: repeatability of words and word combinations that form the term in texts of a given domain.
- Euphony: Ease of pronunciation and absence of undesirable associations

2.1.2 Linguistic Filtering

ATR methods mainly consist of two sub-process. Extraction of candidate terms using linguistic processors and statistical heuristics, followed by candidate ranking and selection. Linguistic filters make use of domain-specific lexico-syntactic patterns to capture term formation and also collocation. These patterns are generally noun phrases and classified into two forms, Closed Filters and Open Filters[Frantzi et al., 2000] [Justeson and Katz, 1995]. Open filters are more permissive and often allow adjectives and sometimes prepositions. Both use Parts of Speech(POS) tags for sequence matching for n-gram extraction, Noun Phrase (NP) chunking and dictionary lookup. Candidate terms are most often normalized and to reduce inflectional forms and stop words are removed.

2.1.3 Statistical Filtering

Statistical processes compute scores for candidate terms to indicate their likelihood of being a term in the domain, consequently classifying the candidates into terms and non-terms based on scores. These ranking algorithms are considered to be the critical and complicated process in ATE methods. The most straightforward statistical filtering that can be used is by removing terms that have a minimal occurrence in the text. This is effectively done by setting the threshold on the frequency of occurrence, which eliminates candidates that are almost impossible to be terms. Some of the researchers have contributed by developing open-source tools that can identify terms on various measure. ATR4S by [Astrakhantsev, 2018], is an implementation of 13 state-of-the-art methods using scala for automatic terminology extraction. The work also evaluates the measures on seven datasets with existing ground truth. Another researcher implemented two versions of JATE [Zhang et al., 2016] using JAVA, which has more than 20 implemented algorithms in its bundle. In the below subsection, few of the statistical measures used in the literature are explained.

2.1.3.1 TF-IDF

TF-IDF a classical method for information retrieval can also be classified as an early feature, It has high values for the terms that frequently occur only in a small number of documents. The measure takes into account the specificity as a function of the term used [Jones, 1972]. The frequency of terms TF in each document is calculated and also

Literature	Main Contribution
[Justeson and Katz, 1995]	Probably one of the first literature to identify grammatical properties in technical terminology. Also presents an open filter (Parts of the Speech pattern of Noun Phrases) to identify Terms
[Kageura and Umino, 1996]	The very first Known surveys on Term Recognition, Introduced Unit-hood and Term-hood, Describes the use of TF-IDF methods, also separates Linguistic and Statistical Class.
[Frantzi et al., 2000]	Introduction of ATR using combination Statistical and linguistic methods. Add on to the open filters of Parts of Speech [Justeson and Katz, 1995] to permit a few more patterns. Introduces C\NC value for statistical filtering
[Pazienza et al., 2005]	Emphasises that terminologies exhibit word associations and describes measure to evaluate them(Dice Factor, Z-test, t-test, χ^2-test, Mutual Information(MI), MI^2, MI^3). Also measuring domain specificity using term frequency, C-value and Co-occurrence.
[Wermter and Hahn, 2006]	Authors show that purely statistics-based measures reveal virtually no difference compared with the frequency of occurrence counts, while linguistically more informed metrics do reveal such a marked difference.
[Zhang et al., 2008]	Experimentally compares five methods TF-IDF, Weirdness, C-Value, Glossex, and Term Extractor for recognizing one-word and multi-word terms. Also infers that the results are skewed to the domain of the dataset.
Braslavskii et al	Compares four different methods for recognition of two-word terms: Term Frequencies, t-test, χ^2-test and likelihood ratio.
[Braslavskii and Sokolov, 2008]	Compares five different methods for recognition of terms of arbitrary structure. Max Len, C-Value, K-factor, Window and AOT. Authors infer that C-Value and K - Factor has high efficiency, while AOT has the lowest efficiency.
[Loukachevitch and Nokel, 2013]	Comparison of methods for recognizing one-word and two-word terms for the problem of thesaurus construction and information retrieval. Infer four different conclusions First: one-word terms can be recognized based on topic models. Second: the combination of several features yields a considerable increase in efficiency as compared to the use of individual features. Third: features based on external corpus offer the most significant increase in efficiency for recognition of two-word terms. Fourth: Word association measures provide no increase with efficiency.
[Fedorenko et al., 2014]	Comparison of two methods based on the combination of features: Voting Algorithm and methods based on supervised machine learning (Logistic and random forest), concluding the second method outperforms the first one.
[Zhang et al., 2016]	Tool implemented in JAVA comprising of 20 Statistical algorithms for ATR
[Astrakhantsev, 2018]	Tool implemented in SCALA comprising of 13 Statistical algorithms for ATR
[Zhang et al., 2018]	Introduces a new method to rank, the first generic method based on the principle of enhancing existing ATR methods by incorporating semantic relatedness in scoring and ranking of candidate terms using page rank algorithm.

Table 2.1: Survey of Literature in Automatic Term Recognition

normalized with the number of terms in the document to avoid inflating values for terms due to its frequent occurrence in the large documents.

$$TF \cdot IDF(t) = TF(t) \cdot \log \frac{1}{TF_d(t)} \tag{2.1}$$

Where,
$TF(t)$ is the number of time the term t appears in the document. It is typically divided by the number of all the of the document to normalize the effect of size of the document.
$TF_d(t)$ is the number of the documents containing the term candidate 't'.

2.1.3.2 Residual-IDF (R-IDF)

RIDF was initially proposed for keywords extraction [Church and Gale, 1999] and then re-used for term recognition [Zhang et al., 2016]. It is based on the assumption that the deviation of observed IDF from the IDF modelled by Poisson distribution is higher for keywords than for ordinary words.

$$R\text{–}IDF(t) = TF(t) \cdot \log \frac{1}{TF_d(t)} + \log(1 - e^{-ATF(t)}) \tag{2.2}$$

Where,
$ATF(t)$ is the average term frequency

2.1.3.3 C-Value\NC-Value

C-Value by [Frantzi et al., 2000], is a domain-specific method used to extract multi-word terms. It aims to get more accurate terms than those obtained by pure frequency method, especially terms that may appear as nested within longer terms, for example, 'Child Pursuant' nested in 'Minor Child Pursuant'. The C-value measure combines linguistic knowledge and statistical information, to obtain termhood measure called c-value.

$$C\text{–}Value(t) = \log_2 |t|(f(t)) - \frac{1}{P(T_t)} \sum_{b \in T_t} f(b) \tag{2.3}$$

Where,
't' is the candidate string,
$f(t)$ its frequency of occurrence in the corpus,
T_t the set of extracted candidate strings that contain a,
$P(T_t)$ the number of these longer candidate terms
$\sum_{b \in T_t} f(b)$ the total frequency by which 't' appears in longer strings.

NC-Value is an extension to c-value, which incorporated context information into term extraction. The method involves three steps.

- Top 10% terms from a domain are recognized and ranked, either by manual selection or by another measure like C-Value. In [Astrakhantsev et al., 2015], the author suggests selecting 200 terms which are recognized using 'Basic'.

- Context words extracted are assigned weights.

$$weight(w) = \frac{t(w)}{n} \tag{2.4}$$

where,
'w' is the context word(noun, verb, adjective or noun phrase)
Weight(w) the assigned weight to the word 'w',
$t(w)$ the number of terms considered.
The purpose of the denominator n is to express this weight as a probability: the probability that the word 'w' might be a term context word. We will elaborate on this point in the following subsection.

- The candidate term list extracted by C-value is re-ranked with the incorporation of context information acquired from the second stage.

$$NC - Value = 0.8 * C - Value(a) + 0.2 * \sum_{b \in C_a} f(b) weight(b) \tag{2.5}$$

where,
'a' is the candidate term,
C_a is the set of distinct context words of a,
b is a word from C_a,
$f_a(b)$ is the frequency of b as a term context word of a
$weight(b)$ is the weight of b as term context word.

2.1.3.4 Basic

Basic is a modification of C-Value for intermediate level (of specificity) terms extraction. Like original C-Value, it can extract multi-word terms only; however, unlike C-Value, Basic promotes term candidates that are part of other term candidates, because such terms are usually served for creation of more specific terms.

$$Basic(t) = |t| \log f(t) + \alpha e_t \tag{2.6}$$

where,
e_t is the number of term candidates containing 't'.

2.1.3.5 ComboBasic

ComboBasic modifies Basic further so that the level of term specificity can be customized by changing parameters of the method.

$$ComboBasic(t) = |t| \log f(t) + \alpha e_t + \beta e'_t \tag{2.7}$$

where,
e'_t is a number of term candidates that are contained in t.

In the literature [Astrakhantsev, 2018], the optimal values for α and β were found to be 0.75 and 0.1 on seven datasets, respectively.

2.1.4 Semantic Relatedness between Terms

Semantic relatedness describes the strength of the semantic association between two concepts or their lexical forms by encompassing a variety of relations between them.[Zhang et al., 2018]. A more specific kind of semantic relatedness is a semantic similarity, where the sense of relatedness is quantified by the 'degree of synonymy' [Weeds, 2003]. Apart from SemRe-Rank[Zhang et al., 2018]. There is no much work found on Semantic relatedness of terms. SemRe-Rank is probably the first generic methods based on principles of enhancing existing ATR methods by incorporating semantic relatedness. SemRe-Rank applies a personalized PageRank process to a semantic relatedness graph of words constructed using the word embedding models[Mikolov et al., 2013a] trained on a domain-specific corpus. The PageRank algorithm [Page et al., 1999] is well-known for its use in computing the importance of nodes in a graph based on the links among them and was initially used to rank web pages.

In the current literature, a novel method has been proposed the word embeddings of the generated terms are used to find the similarities by raking them based on the distanced between them.

2.2 Text Representations

Representing unstructured text as numerical vectors based on the relation between target and contexts is the current state of the art for analysing text in a corpus [Manning et al., 2008]. Also, the distance between numerical vectors can be found more intuitively than the plain text. There are many different ways for learning word embeddings from a corpus; ranging from one-hot encoding [Harris and Harris, 2010] to dense representations [Almeida and Xexéo, 2019] [Camacho-Collados and Pilehvar, 2018]. Intuitively few of them are selected and surveyed to select one for the required task.

2.2.1 Neural Probabilistic Language Model

The model [Bengio et al., 2003] focuses on learning a statistical model of the distribution of word sequences. The model has the vector representations of each word and parameters of the probability function in its parameter set. The objective of the model is to find the parameters that minimize the perplexity of the training dataset. The model eventually learns the distributed representations of each word and the probability function of a sequence as a function of the distributed representations. In this case, even a never-before seen sentence can obtain a high probability if the words in it are similar to those in a previously observed one.

2.2.2 Word2Vec

Tomas Mikolov et al. [Mikolov et al., 2013a] developed log-linear models which observed significant improvements in accuracy and performance at a much lower computational cost. Besides using word offset technique using simple vector algebraic operations, it was found that word representations go beyond simple syntactic regularities. For example,

vector(King) - *vector*(Man) + *vector*(Woman) results in a vector that is closest to the vector representation of the word Queen.
The literature proposes two new log-linear models for learning distributed representation of words minimizing the computational complexity.

1. Continuous Bag of words(CBOW): The architecture has three layers; an input layer, projection layer and output layer. The input layer is comprised of the sum one-hot vector of the words within the context window of the target word. The output layer is a one-hot representation of the target word. The projection layer weights are trained to map these relations using the words in the corpus. Finally, the projection layer will be the vector representation of the output word in a lower the dimension.

2. Continuous Skip Gram Model: The architecture is similar to the CBOW model. In the skip-gram, the context one-hot representation of words is mapped at the output given the one-hot representation of the target word.

The computation is still expensive and makes it impractical to scale up to large vocabularies or large training corpora. This problem is solved by limiting the number of output vectors that will be updated for an epoch [Mikolov et al., 2013b]. This is achieved by hierarchical softmax, negative sampling asynchronous Stochastic Gradient Descent.

2.2.3 Global Vectors (GloVe)

Unlike the word2vec models, the Glove model [Pennington et al., 2014] learns word vectors from the term co-occurrence matrix. A co-occurrence matrix is of size $[V * V]$ where V is the vocabulary size of the corpus. Each entry corresponds to the frequency of the word within a window of context. The resulting co-occurrence matrix is transformed to a lower dimension by matrix factorization methods.

Word vectors estimated using GloVe are conceptually similar to those derived from word2vec but uses an underlying count-based model, rather than word2vec's prediction-based model. Because GloVe typically computes statistics over larger context windows than word2vec, it permits capturing longer-term dependencies, although the order of those dependencies will be lost. Empirically, no clear advantage has emerged for either word2vec or GloVe, as the overall performance depends on various factors, including the type of data and evaluation task being considered.

2.2.4 Non-Parametric Multi-Sense Skip Gram(NP-MSSG)

MSSG [Neelakantan et al., 2015] Another extension of word2vec skip-gram model. Multiple embeddings are learnt for the same word each having different meaning resulting in sense discrimination. Two variant models have been proposed in the literature. The most important property that differentiates the model from above is that the vectors provide word sense disambiguation. Multiple vectors are learned per word, which gives the representation of senses.

1. MSSG: MultiSense Skip Gram model learning fixed set of senses for a word across the corpus.

2. NP-MSSG: Non-parametric version of the above model to learn varying number of senses per word type.

2.2.5 FastText

Developed by Facebook research [Bojanowski et al., 2017] [Joulin et al., 2016b] [Joulin et al., 2016a] is an extension of word2vec with enriched embeddings for subword information. In this approach, the embeddings are generated for n-grams along with the word. It can be trained with either using skip-gram(SG) or Collective Bag of words(CBOW). The N-grams are created by moving a window of generally with a size of 3 to 6 characters. These n-grams are used to train a log-linear model. The final word embedding is calculated as the sum of n-gram embeddings that constitute the word. The n-gram information allows sharing the representation across words, thus handling, out- of- vocabulary terms(OOV).

For example, the tokens *reality* and *terms* will be split to subwords with window 3 in the following way $< re, rea, eal, ali, lit, ity, ty >$ and $< te, ter, erm, rms, ms >$. The algorithm trains embeddings for all these words and also the token terms. Now, suppose if we look for a new word $'realms'$ which is currently absent in the model vocabulary. The model uses the embeddings of the subword information *rea*, *eal* and *ms*, adds their vector representations to give a fairly good representation for '*realms*'. As per the authors, these representations have comparable similarities with their synonyms.

FastText replaces the softmax over labels with a hierarchical softmax. Here each node represents a label. This reduces computation as label probabilities are not processed. The limited number of parameters reduces training time.

2.2.6 ELMo

ELMo - Embedding from Language Models [Peters et al., 2018] is a contextualized word and character-level embedding. Instead of using a fixed embedding for each word, ELMo looks at the entire sentence as it assigns each word an embedding. It uses several stacked bi-directional recurrent neural networks (LSTM) trained on a specific task to create the embeddings. Since it uses a bidirectional architecture, the embedding is based on both the next and previous words in the sentence.

The input to the biLM is computed from characters rather than words; it captures the inner structure of the word. For example, the biLM will be able to figure out that terms like beauty and beautiful are related at some level without even looking at the context they often appear in. ELMo word representations take the entire input sentence into the equation for calculating the word embeddings. Hence produced multiple word embeddings per single word for different scenarios. Higher-level layers capture context-dependent aspects of word embeddings while lower-level layers capture model aspects of syntax.

2.2.7 BERT

BERT [Devlin et al., 2018] - Bidirectional Encoder Representations from Transformers is another contextualized word representation model based on a multilayer bi-directional

transformer-encoder, where the transformer neural network uses parallel attention layers rather than sequential recurrence. BERT is pre-trained on two unsupervised tasks: (1) a 'masked language model, where 15% of the tokens are randomly masked (i.e., replaced with the "[MASK]" token), and the model is trained to predict the masked tokens, (2) a 'next sentence prediction' (NSP) task, where the model is given a pair of sentences and is trained to identify when the second one follows the first. This second task is meant to capture more long-term or pragmatic information.

The comparison for all the embedding models mentioned above are tabulated below.

Model & Author	**Objective**	**Architecture**	**Notes**
Neural Probabilistic language Model (Bengio Et al . 2003)	Learn Joint probability function of the sequence of words in a language, alleviate the curse of dimensionality.	Neural Network with Tanh activation and backpropagation, coupled with Hierarchical Softmax	First of the neural network language model. Not efficient with Scaling.
Word2Vec (Mikolov et al 2013 a,b)	Increasing the efficiency of learning embeddings. Given the context-window, predict target words(CBOW) or vice versa(SG).	Log-linear Model, CBOW/Skip Gram models, coupled with Hierarchical SoftMax.	Word vectors can be somewhat meaningfully combined using just simple vector addition.
Word2Vec (Mikolov et al 2013 c)	Several extensions that improve both the quality of the vectors and the training speed. Subsampling of the frequent words to obtain speed up.	Log-linear Model, CBOW/Skip Gram models, coupled with Negative Sampling.	SGNS (skip-gram with negative sampling), the best performing variant of Word2Vec.
GloVe (Pennington et al.)	Improve Semantic similarities with respect to the global context, using the insight that co-occurrence ratios, rather than raw counts.	Log-linear model trained on statistics of word occurrences in a corpus.	Does not cover ambiguity of words.
MSSG (Neelakantan et al.)	Learn embeddings for context and sense jointly	Neural Network: Extension of the skip-gram model.	Only use local contexts to induce sense representations.
Topical Word Embedding [Liu et al., 2015] (Liu et al.)	Employ topic models with word embeddings. Include complicated correlations among words as well as their contexts 3.	Neural Network Models: Topics obtained by LDA and Gibbs sampling to iteratively assign latent topics.	Fixed number of senses per word.
FastText (Bojanowski et al.)	Model embeddings for out-of-vocabulary words(OOV).	Log-linear model, trained with character n-grams, hierarchical softmax.	Enriching Word Vectors with Subword Information. Use Hashing to improve speed.
ELMo (Peters et al.)	Contextualized word- and character-level embedding.	Neural Network: Multilayer bidirectional LSTM.	Higher level of LSTM States used for Contextualized representations.
BERT (Jacob et al.)	Single model to achieve SOTA results in various NLP tasks.	Neural Network: multi-layer bidirectional Transformer pre-trained BERT model.	Fine-tune to use the model for various tasks.

Table 2.2: Table of comparison for different embedding models

2.3 Similarity Measures

After the text is converted into points in vector space, we need to compute the similarity between text to find the related terms. Existing work [Gomaa et al., 2013] on text similarities are analysed to find an appropriate measure.

Sting based similarity measures are categorised into two, Character-based and Term based.

2.3.1 Character-Based Similarity Measures

In Character-based measures, the comparison is limited to the character by character. These measures fail to compare the semantic nature of the words. Words like 'Talk' and 'Speak' are incredibly similar semantically, but none of the characters are the same in both.

1. Longest Common Substring (LCS)
2. Damerau-Levenshtein
3. Jaro
4. Jaro–Winkler
5. Needleman-Wunsch
6. Smith-Waterman
7. N-gram

2.3.2 Term-Based Similarity Measures

This category gives the statistic of the distance between words according to the information gained.

2.3.2.1 Manhattan Distance

Also known as Block boxcar distance, absolute value distance, L1 distance and city block distance. It computes the distance that would be travelled to get from one data point to the other if a grid-like path is followed.

2.3.2.2 Dice's Coefficient

It is defined as twice the number of common terms in the compared strings divided by the total number of terms in both strings.

2.3.2.3 Euclidean Distance

L2 distance is the square root of the sum of squared differences between corresponding elements of the two vectors.

2.3.2.4 Jaccard Similarity

Computed as the number of shared terms over the number of all unique terms in both strings.

2.3.2.5 Word Movers Distance

The WMD distance measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document.

2.3.2.6 Cosine Similarity

It is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them.

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{2.8}$$

The list of Text-Similarity measures available in the literature are tabulated below.

Category	**Description**	**Measures**	
String Based Similarity	Measures operate on string sequences and character composition. Measures similarity or dissimilarity (distance) between two text strings for approximate string matching or comparison.	Character Based	Longest Common SubString (LCS)
			Damerau-Levenshtein
			Jaro
			Jaro–Winkler
			Needleman-Wunsch
			Smith-Waterman
			N-gram
		Term Based	Block
			Cosine
			Dice's
			Euclidean
			Jaccard
			Matching
			Overlap
Corpus-Based Similarity	A Measure that determines the similarity between words according to information gained from large corpora.	Hyperspace Analogue to Language (HAL)	
		Latent Semantic Analysis (LSA)	
		Explicit Semantic Analysis (ESA)	
		Pointwise Mutual Information - Information-Retrieval (PMI-IR)	
		Normalized Google Distance (NGD)	
		Extracting Distributionally similar words-using Co-occurrences (DISCO)	
Knowledge-Based Similarity	Measures that bases on identifying the degree of similarity between words using information derived from semantic networks like WordNet.	Similarity	Resnik (res)
			Lin (lin)
			Jiang & Conrath (jcn)
			Leacock & Chodorow (lch)
			Wu & Palmer (wup)
			Path Length (path)
		Relatedness	St.Onge (hso)
			Lesk (lesk)
			Vector

Table 2.3: Text Similarity Measures

Chapter 3

Methodology

3.1 Methodology Overview

The below section presents the methodology used to automatically recognizing terms (ATR) from a corpus and eventually building up a rationale for a novel process to query terms which are semantically and contextually related. The complete architecture is represented in Figure 3.2. ATR can be perceived to be an unsupervised task as the recognition is ambiguous depending on the domain. As discussed in the previous section, there are a several of state-of-the-art methods, but none can be argued to be best suited for a particular setting. Furthermore, the validation is also dependent on the discretion of a specialist in the area like an ontologist.

The main components of the system are Data Preprocessing, tokenization, Parts of Speech tagging (POS), linguistic filtering, statistical filtering, feature extraction, similarity calculation, context/subject recognition aided by information from the external knowledge base.

3.2 Automatic Terminology Recognition

3.2.1 Data Collection

The text for the dataset in question needs to be first extracted in a format that is recognizable for the tokenizer. If it is a huge dataset, then methods to stream without opening the file, need to be explored. The detailed extraction of Illinois dataset will be mentioned in the implementation section 5.1.1. Also, the datasets for analysis of ATR are used from pre-extracted text files as provided by [Astrakhantsev, 2018].

3.2.2 Data Tokenization

The Initial steps towards data preprocessing are tokenization. The text extracted are split at each sentences breaks, and then each word in the sentence is further split at word level by using white spaces. List of words in a sentence is enclosed in a list data structure, and all sentences of a CD are enclosed as a document list.

Input string sequence: *'JUSTICE BILANDIC\ndelivered the opinion of the court:\nThe appellant, the Illinois Department of Public Aid (the Department), filed a motion to intervene in a dissolution action pending between Larry and Lynn Lappe in the circuit court of Madison County.'*

Tokenized String: ['JUSTICE', 'BILANDIC', 'delivered', 'the', 'opinion', 'of', 'the', 'court:', 'The', 'appellant,', 'the', 'Illinois', 'Department', 'of', 'Public', 'Aid', '(the', 'Department),', 'filed', 'a', 'motion', 'to', 'intervene', 'in', 'a', 'dissolution', 'action', 'pending', 'between', 'Larry', 'and', 'Lynn', 'Lappe', 'in', 'the', 'circuit', 'court', 'of', 'Madison', 'County.']

3.2.3 Data Preprocessing

The standard process of text pre-processing for most language tasks follows the below steps.

- Uniform case conversion
- Numbers to word conversion
- Removing special characters
- Expanding contractions
- Removing stop words, sparse terms, non-words, url's, and frequent terms.
- Text canonicalization by lemmatization and stemming

Contrary to the above on the current text, minimalistic text pre-processing is performed. As the downstream process is to tag parts of speech for each token, the morphology of text needs to be preserved for precise POS tagging. The detailed reasons will be mentioned in the next tagging section. Post tokenization the text containing non-word character sequences are removed and finally stripped off any special characters. This feat is achieved using a sequence of a regular expression.

Preprocessed Tokens: ['JUSTICE', 'BILANDIC', 'delivered', 'the', 'opinion', 'of', 'the', 'court', 'The', 'appellant', 'the', 'Illinois', 'Department', 'of', 'Public', 'Aid', 'Department', 'filed', 'a', 'motion', 'to', 'intervene', 'in', 'a', 'dissolution', 'action', 'pending', 'between', 'Larry', 'and', 'Lynn', 'Lappe', 'in', 'the', 'circuit', 'court', 'of', 'Madison', 'County']

3.2.4 Parts of Speech Identification

POS identification is a "supervised learning problem". The tokenized words are identified for their corresponding parts of speech in the sentence. The idea here is to identify noun phrases in the downstream task, hence to identify multi-word terms knowing the parts of speech of each token/word is crucial. In the next chapter, we detail on the tagger used

Tag	Tag Description
$	dollar
”	closing quotation mark
(	opening parenthesis
)	closing parenthesis
,	comma
–	dash
.	sentence terminator
:	colon or ellipsis
CC	conjunction, coordinating
CD	numeral, cardinal
DT	determiner
EX	existential there
FW	foreign word
IN	preposition or conjunction, subordinating
JJ	adjective or numeral, ordinal
JJR	adjective, comparative
JJS	adjective, superlative
LS	list item marker
MD	modal auxiliary
NN	noun, common, singular or mass
NNP	noun, proper, singular
NNPS	noun, proper, plural
NNS	noun, common, plural
PDT	pre-determiner
POS	genitive marker
PRP	pronoun, personal
PRP$	pronoun, possessive
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	particle
SYM	symbol
TO	to as preposition or infinitive marker
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle or gerund
VBN	verb, past participle
VBP	verb, present tense, not 3rd person singular
VBZ	verb, present tense, 3rd person singular
WDT	WH-determiner
WP	WH-pronoun
WP$	WH-pronoun, possessive
WRB	Wh-adverb
“	opening quotation mark

Table 3.1: Parts of Speech tags in NLTK

and the statistics from the dataset. Below is the example of how the tagged tokens are perceived. The tag descriptions are available in Table 3.1.

POS tagged Tokens: [('JUSTICE', 'NNP'), ('BILANDIC', 'NNP'), ('delivered', 'VBD'), ('the', 'DT'), ('opinion', 'NN'), ('of', 'IN'), ('the', 'DT'), ('court', 'NN'),....('Lynn', 'NNP'), ('Lappe', 'NNP'), ('in', 'IN'), ('the', 'DT'), ('circuit', 'NN'), ('court', 'NN'), ('of', 'IN'), ('Madison', 'NNP'), ('County', 'NNP')]

3.2.5 Linguistic Filtering

As quoted by [Frantzi et al., 2000], *"It would be 'very desirable' for a method to be able to extract all types of terms (e.g. noun phrases, adjectival phrases, verbal phrases.)"*. No such approach has been followed yet in the literature of ATR. Directly applying statistical filters for ATR results in undesirable terms such as "is", "a".

Extracting terminologies from text is an NP-Complete problem. Terminologies can be constituted from a single word or using a sequence of words (multi-word phrase). Assuming the word length of a term $|T|$ to not exceed the number of words in a sentence n, The total number of possible terms in a sentence is given by the formula from the arithmetic progression of natural numbers $n(n+1)/2$, also called as the triangular number. As we can see the number of terms to be sifted increases by n each time. With a large number of sentences in the solution a brute force solution to the problem is seen as NP-complete (non-deterministic polynomial time), where the solution can be verified quickly, but having no way to find a solution quickly. That is, the time required to solve the problem using any currently known algorithm increases rapidly as the size of the problem grows. Hence

to avoid finding candidates in polynomial time, the syntactic feature of a text "Nominativeness" is exploited to heuristically determine word sequences having possibilities of forming a candidate for a term. One such entity that can provide information on the feature is "Parts of Speech(POS).

3.2.5.1 POS Patterns for Candidate Extraction

As mentioned in the previous section, terminologies are made-up of single-word and multi-word candidates. Single-word candidates are filtered out using their POS, an example shown in Figure3.1. Nouns have a maximum likelihood of being categorised as terms. Whereas for multi-word candidates Noun Phases which are formed from a combination of terms following specific sequences made up of adjectives, nouns, and rarely including prepositions and determinants have the maximum probability of categorising those sequences(like noun phrases, adjectival phrases, verbal phrases). The Noun Phrases in the text are detected using Regular Expression patterns defined by multiple allowed sequences of POS tags. Defining such sequences usually are empirical to the dataset or domain.

As per literature [Frantzi et al., 2000], Since terms consist of nouns, adjectives and rarely prepositions, set of linguistic features are built that can accept these types of terms. The choice of the linguistic filter affects the precision and recall of the output list. Various researchers have used different patterns. These filters are categorised into two types.

- 'Closed' Filter which is strict about the sequences of POS it permits, it results in a positive effect on the precision but negatively effects the recall. E.g. a filter which uses only noun sequences (like $Noun + Noun$), it produces high precision since noun sequences in a corpus are most likely to be a term, but negatively effects recall as there are many noun sequences found in a dataset.

- 'Open' Filter one that permits strings with multiple combinations of POS. Such filters have a reverse effect of prior. These patterns include other variations of POS present in Noun Phrases. An example of such a filter is mentioned in [Justeson and Katz, 1995].

  $$((Adj|Noun) + |((Adj|Noun) * (NounPreposition)?)(Adj|Noun)*)Noun$$

  Whereas in [Frantzi et al., 2000] add two more filters to permit more terms, to balance recall and precision.

  1. $Noun + Noun$
  2. $(Adj|Noun) + Noun$
  3. $((Adj|Noun) + |((Adj|Noun) * (NounPreposition)?)(Adj|Noun)*)Noun$

- Personalized Filter: Linguistic filters are generally domain-dependent and empirically designed. Set of filters mentioned below, has been empirically designed to retrieve terms in Illinois Case Dataset. Exploiting the depth of tags a more specific

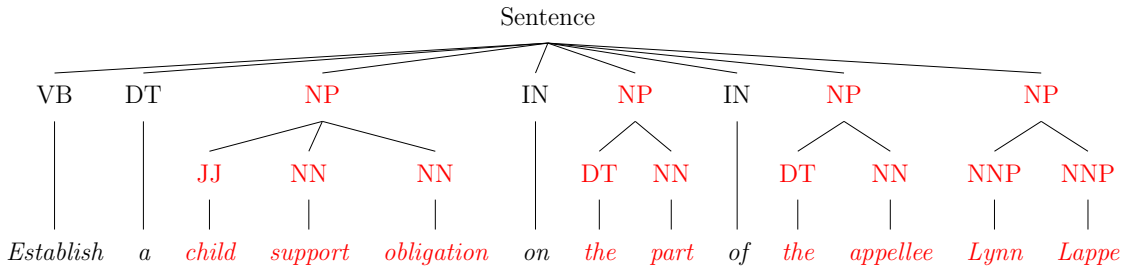

Figure 3.1: Noun Phrases based on POS Tags

design is achieved, and in further sections, we show it does better than the above filters.

These sequences are defined in order to extract the longest probable sequence starting from the first in the pattern order. This order limits the nesting of the candidates from the same sequence. The whole logic is implemented in python using the NLTK library. The words are extracted sequentially corresponding to the patterns defined. The group of these words form the candidate for the term. The pattern is empirical and is designed, keeping in mind the Illinois Dataset. In a biomedical corpus, the terminologies are longer and may also contain verbs and prepositions between nouns. The tag abbreviations are mentioned in Table3.1

1. $\{< NNP > 1,3 < NN? >?\}$
   $ProperNouns(1to3) + (Proper|Common|PluralNoun)$
   Eg. 'JUSTICE BILANDIC'
2. $\{(< JJ > | < NN >) < NN? > 1,3\}$
   $(Adjective|CommonNoun)+(ProperNoun|CommonNoun|PluralNoun)(1or3)$
   Eg. 'circuit court'
3. $\{< DT >? < JJ > * < NN >\}$
   $Determiner(1or0) + Adjective(0orany) + CommonNoun$
   Eg. 'the divorced couple's'
4. $\{< DT >? < NN > +\}$
   $Determiner(1or0) + CommonNoun$
   Eg: 'a petition'

Example Chunked tokens :['JUSTICE BILANDIC', 'opinion', 'court', 'appellant', 'Illinois Department', 'Public Aid Department', 'motion', 'dissolution action', 'Larry', 'Lynn Lappe', 'circuit court', 'Madison County']

3.2.6 Statistical Filtering

Although the candidates recognized from the linguistic filtering are semantically valid, they still need to be separated to adequately represent the textual information of the domain for underlying knowledge elicitation tasks. This can be achieved by filtering the candidates using statistical measures.

Depending on the features of terms, the statistical measures are grouped into different categories by terminology researchers[Astrakhantsev et al., 2015][YUAN et al., 2015] [Heylen and De Hertog, 2015][Kageura and Umino, 1996]. Summarising the existing literature, a conclusion is reached to evaluate the measures on terms which best exhibit the property of 'Unithood' [1] and 'Termhood' [2]. Unithood and termhood refer to the qualities of terms, or as Nakagawa calls it "two essential aspects of the nature of terms" [Nakagawa, 2001]. For all the statistical measures

3.2.6.1 TF-IDF

The frequency of terms TF in each document is calculated and also normalized with the number of terms in the document to avoid inflating values for terms due to its frequent occurrence in the large documents. TF-IDF score for each term in a document is calculated and saved in a dictionary. The terms are selected based on a threshold that is decided empirically.

3.2.6.2 Residual IDF

It is based on the assumption that the deviation of observed IDF from the IDF modelled by Poisson distribution is higher for keywords than for ordinary words. The terms are selected based on a threshold that is decided empirically.

3.2.6.3 C-Value/NC- Value

The C-Value across all the terms in all the documents are calculated. Hence for a corpus, we obtain unique values for each term. The values are sorted and then filtered based on a threshold that is decided empirically, taking into account the chunk patterns for Noun phrases. The measure requires an α value which is a smoothening parameter for one-word terms. The literature in [Lossio-Ventura et al., 2013] suggests to use $\alpha = 1$, but as per experiments in [Astrakhantsev, 2018], it is found that better results are achieved α is set to 0.1.

3.2.6.4 Basic/ComboBasic

The ComboBasic and Basic have additional coefficient α, βthat needs to be tuned. In [Astrakhantsev, 2018], combination of coefficients from $\alpha = \{0, 0.1, 0.5, 0.75, 1\}$ and $\beta = \{0, 0.1, 0.5, -0.1, -0.25, -0.5\}$ were tested on 6 terminology datasets and it was decided that $\alpha = 0.75, \beta = 0.1$ gives the top 'F1' score for the ATR task. Hence the same parameters are used for the analysis. This measure gives the top interpretation of the contextual terms from the database.

[1] Unithood refers to a degree of strength or stability of syntagmatic combinations or collocations.

[2] Termhood refers to a degree of linguistic unit. It considers a term as a linguistic unit representative for the document content. The higher the termhood is, the higher capability of distinguishing different domains the terminology has[Zhang and Wu, 2012]

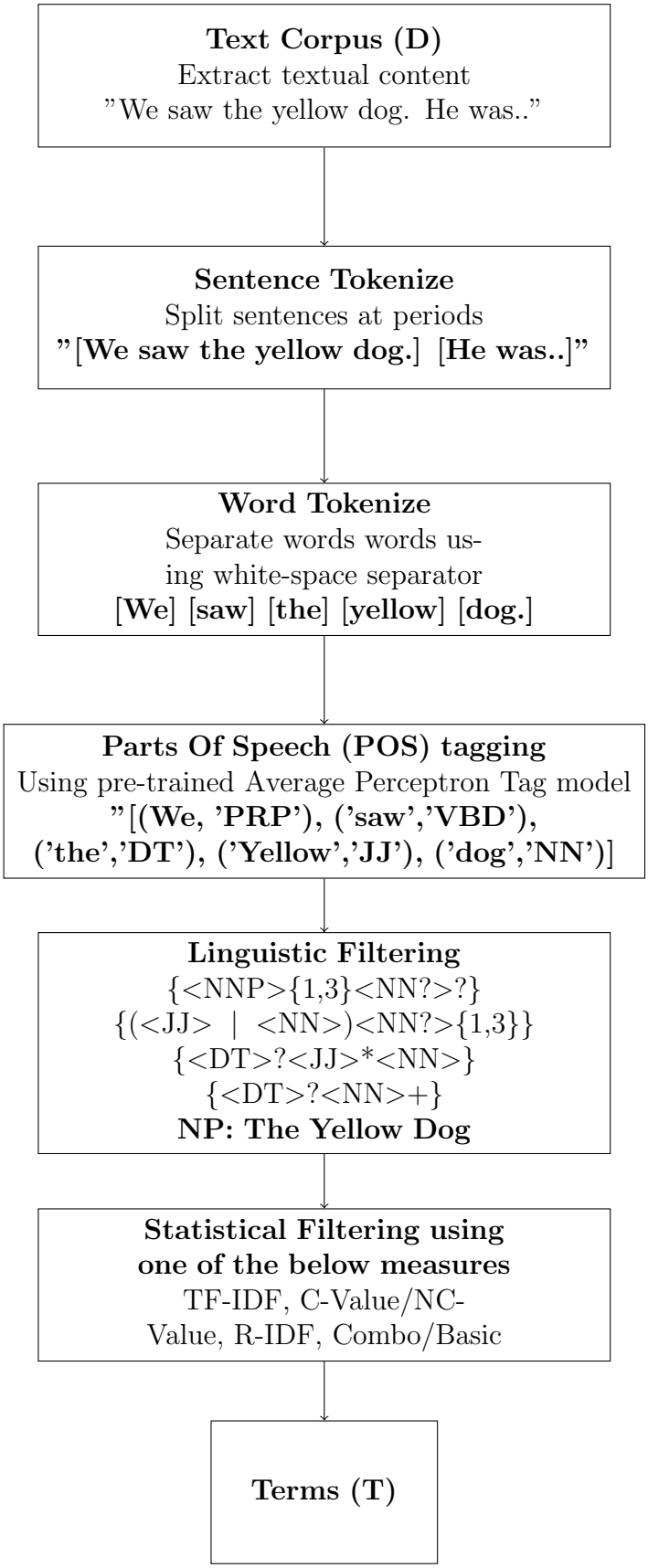

Figure 3.2: Model for Automatic Term Recognition

Algorithm 1 Term Recognition Model

Input: Dataset
Output: Candidate Terms (T)

function SentTokenizer(*documents*)
  Sentences ← Split(*documents*,′ .′) ▷ Split at periods
  sentences ← PreProcess(sentences)
  return sentences ▷ return sentences
end function
function PreProcess(*sentences*)
  sentences ← RemoveStopWords(*sentences*) ▷ remove Stopwords
  sentences ← RemoveSpecialCharacters(*sentences*) ▷ Remove Special Characters
  return sentences ▷ return sentences
end function
function WordTokenizer(*Sentence*)
  token ← Split(*Sentence*,′ _′) ▷ Split at White spaces
  return tokens ▷ return token
end function
function POS_Tagger(*token*)
  token ← PreTrainedPOSModel(*token*) ▷ Average Perceptron Model
  return tagged_tokens ▷ return tagged tokens
end function
function NounPhraseChunker(*tagged_tokens*)
  grammar ← *regex_patterns*(adj+Nouns, Noun+Noun, adj+prep+noun)
  noun_phrases ← Parse(*grammar*, *tagged_tokens*) ▷ extract noun phrases
  return noun_phrases ▷ return noun phrases
end function
function StatisticalFilter(*noun_phrases*)
  candidate_terms ← tf_idf(noun_phrases)
  return candidate_terms ▷ return candidate terms
end function
while !*EOF*(*Corpus*) **do** ▷ Iteration Count
  documents ← StreamNext(*Corpus*) ▷ Stream one object at a time from dataset
  Sentences ← SentenceTokenizer(*documents*(*D*))
  for *sentence* ← *Sentences* **do** ▷ Loop each sentences
    tokens ← WordTokenizer(*sentence*) ▷ Generate tokens
    for *token* ← *tokens* **do** ▷ Loop each sentences
      tagged_tokens ← POS_Tagger(*token*) ▷ Generate tokens
      noun_phrases ← NounPhraseChunker(*tagged_tokens*) ▷ Generate noun phrase
      *term_candidates*T← StatisticalFilter(*noun_phrases*) ▷ Generate terms
    end for
  end for
end while

3.3 Semantically Relatedness of Terms

This section describes the process used to show the contextual relationship between terms. The process leading to the mining of terms having different semantic relations is shown in Figure3.3 and Figure3.4. To measure term relations, the extracted terms need to be featurised into numbers, as numbers are convenient to measure. There are several methods to achieve this, and few of them are discussed in the earlier chapter. Concluding from the current state of the art, word embeddings, especially trained using skip-gram algorithms provide superior semantic results[Wang et al., 2019][Mikolov et al., 2013a]. Word embeddings map a word to a point in n-dimensional vector space.

Traditional Embedding algorithms encode single word tokens to a vector space. Semantic operations can undoubtedly benefit from exploiting features extracted from single-word terms, but a domain-specific knowledge retrieval consists of information gained using multi-word terms or collocations. Embedding algorithms can be tweaked to handle multi-word terms. Word2vec algorithms can be trained using n-grams from the corpus as tokens. This process can become computationally expensive on a large corpus as the number of vectors to be trained increases by a factor of $n(n+1)/2$. Moreover, most of the n-grams trained remain unused as they have no morphological, semantic meaning in the standard text. Another method is to just add the corresponding word vectors constituted in the term. FastText trains character n-grams; hence it can produce out of word vocabulary word vector by averaging the character n-grams constituted in the texts.

Embeddings models can be created using the text - corpus in question. Also, a pre-trained embedding model trained on a large corpus can be used to get word vectors[3]. Word embedding model is created by training embedding algorithms[4] on the text that is used to extract terms. Parallelly the results are compared with word embeddings from pre-trained Wikipedia model.

3.3.1 Word Embedding Training

FastText is a library for learning word embedding and text classification developed by Facebook based on the works of literature by Bojanowski [Bojanowski et al., 2017] and Joulin et al [Joulin et al., 2016b]. The embeddings are generated by training the concatenated case decisions file. Although our goal is to find relatedness only between extracted terms, the algorithm is still trained on the whole corpus. The terms do contain syntactical information within their units, but the semantic information would be missing due to the absence of text that failed to make it in the terminology. Hence the embeddings are trained by tokenizing the text normally and later generalised by agglutinating the tokens consisted in the terms.

Initially, word2vec algorithms were designed to give word vectors for words present only in the training text vocabulary. In the recent past, specific algorithms can provide vectors to words which are not in its training vocabulary. Few word2vec algorithms with their properties are mentioned in Table3.2. The model trained on fastText has another benefit as it can generalise out of vocabulary words using character n-grams it

[3]Word vector and word embeddings have the same meaning, and can be used interchangeably
[4]Skip-gram variant in this case

is trained on. Since terminologies are not trained exclusively for embeddings, the model can approximate vectors which exhibit comparable semantic properties.

Model	Handles OOV	Sub-word info	Order of dim.
Co-Occurrence Matrix			V2
MSSG and NP-MSSG			D
CBOW (Word2Vec)			D
Skip-gram (Word2Vec)			D
GloVe			D
FastText	✓	✓	D
ELMo	✓	✓	D
Probabilistic FastText	✓	✓	D
BERT	✓	✓	D

Table 3.2: Word embeddings Model Characteristics

The comparison of a few models having using sub-word information and Out of Vocabulary Vector.
V is vocabulary size, and D is the size of the vector.

The text streamed from CD's is concatenated into a file. To get maximum semantic features, no preprocessing is done on the text, and the file is used to train the fastText algorithm. The algorithm is trained with multiple variations of hyper-parameters and also with another variation by normalizing the text.

3.3.2 Term Embeddings

The Term embeddings are generated from the model using two methods.

- Summing the tokens that make the terms
  Eg: Embedding of 'Public Aid Department' = sum of ($V[Public]$+$V[Aid]$+$[Department]$)
- Directly extracting the word vectors from the model which used the character n-grams that make the terms.
  Eg: 'Public Aid' = sum($< Pub >< ubl >< bli > .... < c\ a >< \ ai >< aid >$)

Either way, the final embedding for a term will be a vector with n-dimensions. Vectors for each term are extracted and saved in an array of vectors. The final model results in a matrix of shape $n * |T|$

3.3.3 Discerning Similarity Information

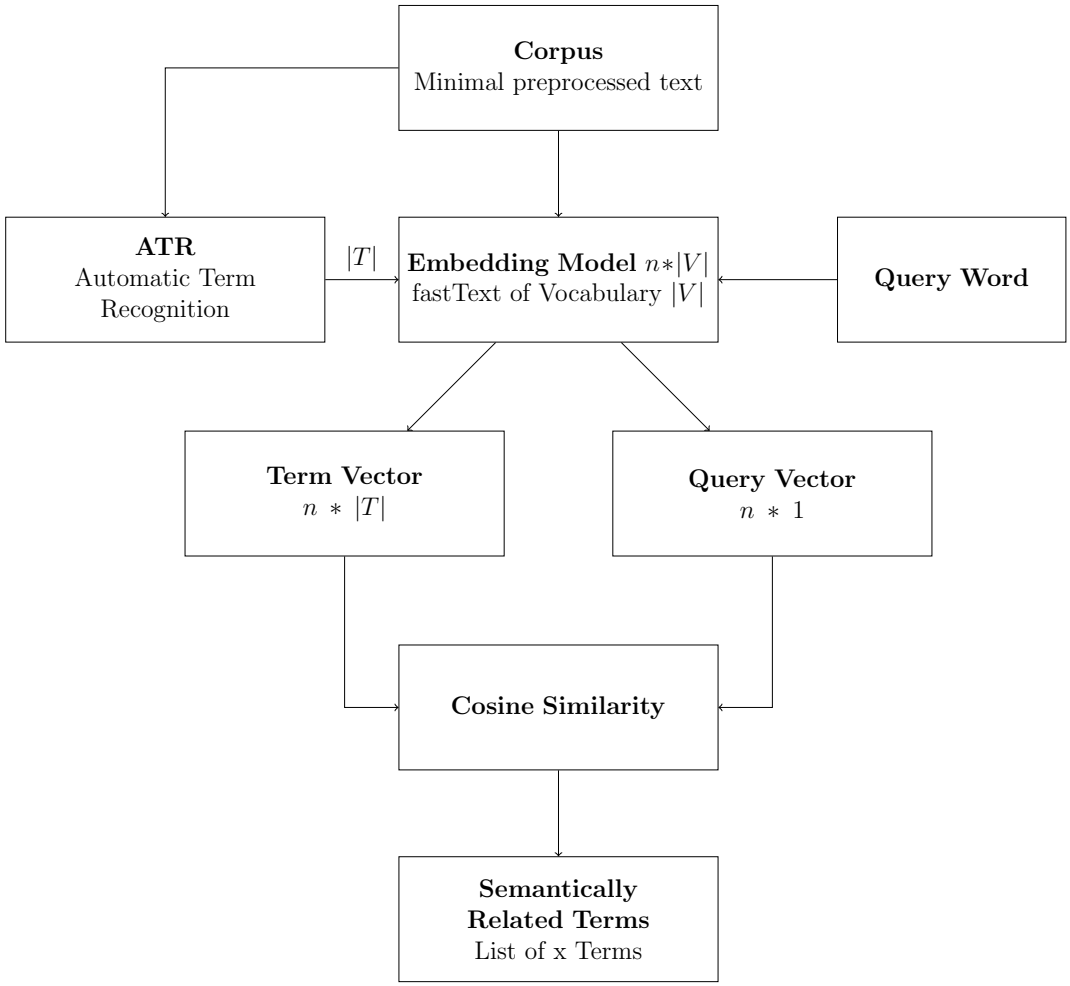

Figure 3.3: Model for generating semantic relations

To obtain semantic relations within the terms or with the external concept. For this task, the word vectors can be useful to find relations between them. One of the most widely used measure for similarity is the 'cosine similarity'. Cosine distance is given by calculating the cosine of the angle between the vectors. There are also other measures like Word Movers Distance measure(WMD)[Kusner et al., 2015]. WMD is a text variant of EMD. Although the measure is a suitable distance measure, the semantic similarity is better captured using cosine distance. Also, the computation is also slow as the average time of solving WMD is about $O(p^3 logp)$. Whereas with cosine distance can be calculated by vector operations. The architecture is detailed in the Figure3.3, and the algorithm used is mentioned in 2.

Algorithm 2 Semantic Relation Model

Input: Corpus Text, Query
Output: Similar Terms

1: $documents \leftarrow$ StreamNext($Corpus$) ▷ Stream one object at a time from the dataset
2: $Embedding\_Vectors \leftarrow$ Embedding_Algorithm($documents$)
3: $CandidateTerms \leftarrow$ ATR($documents$)
4: $Term\_Vectors \leftarrow Embedding\_Vectors[CandidateTerms]$
5: $Query\_Vector \leftarrow Embedding\_Vectors[Query]$
6: $Similar\_Terms \leftarrow$ Cosine($Query\_Vector, Embedding\_Vector$)

3.3.4 Context Information Mining

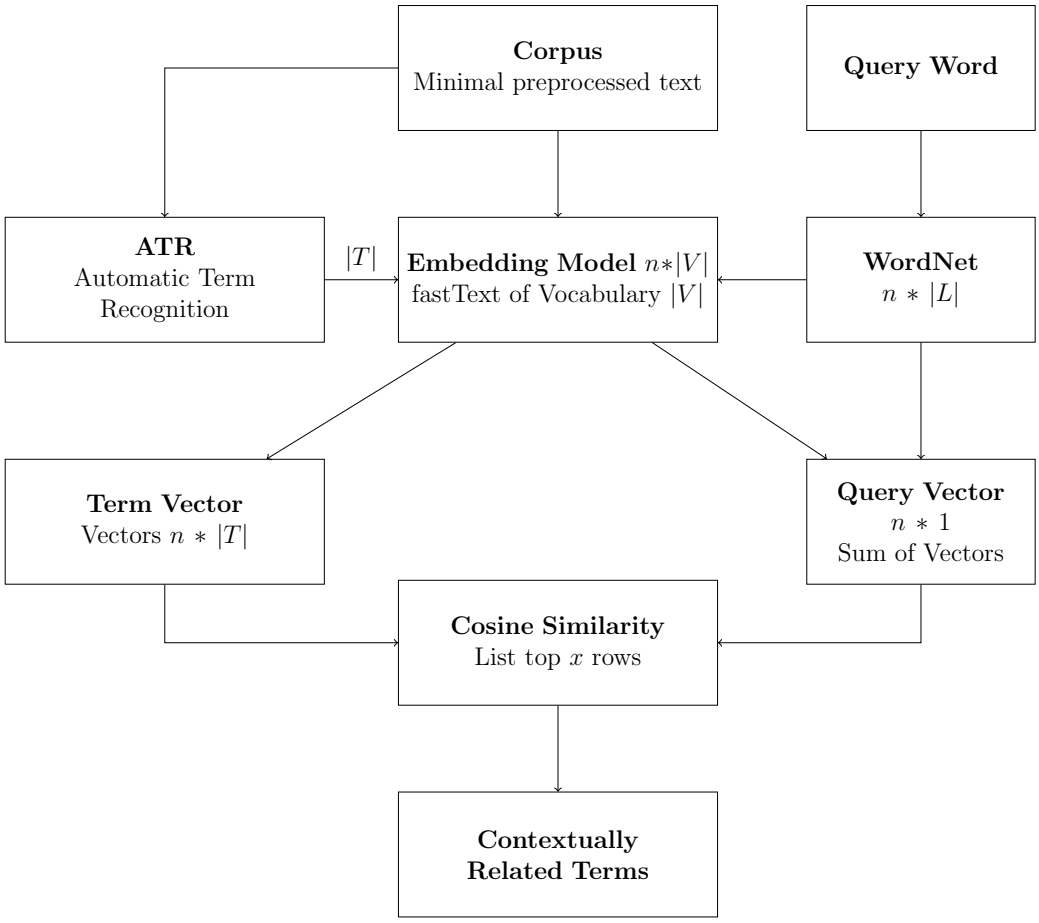

Figure 3.4: Model for Generating Contextual relations

Information at the level of cognition of context information from the term is the tricky part of this system. The context of a term is ambiguous and is pre-existing dependent knowledge of the domain. The knowledge source can be an information graph like an

appendix of a book which contains terminologies listed below a topic. Hence the terms listed below a topic be enlisted as a subject for the terms. In this literature, the subject information is emulated by augmenting information from an external knowledge base like the 'WordNet'. WordNet is a lexical database of English where nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms. The architecture is shown in Figure 3.4.

Context or subject information of a term is parallel to super-subordinate relation like hyperonym and homonymy of words in taxonomy graph. These relations are used to identify context terms in WordNet. Later these terms similar to context terms are listed for validation.

WordNet is a knowledge-base with 150K words; hence there is a high probability of finding words in the database. For the words out of vocabulary, most similar candidates in WordNet are used to find super- subordinate information. Similar candidates are selected using the following process.

- The vocabulary of words in WordNet are vectorised using the fasttext model.
- Then the vector for the query is generated using the character n-grams in the fasttext model.
- This query vector is used to find the closest available words in WordNet vocabulary.
- The super- subordinate vectors of the closest available words are summed up and then looked into terminology matrix using cosine distance

Algorithm 3 Context Relation Model

Input: Corpus Text, Query
Output: Contextual Terms

$documents \leftarrow$ StreamNext($Corpus$) ▷ Stream one object at a time from dataset
$Embedding\_Vectors \leftarrow$ Embedding_Algorithm($documents$)
$CandidateTerms \leftarrow$ ATR($documents$)
$Term\_Vectors \leftarrow Embedding\_Vectors[CandidateTerms]$
$WorNet\_Vectors \leftarrow Embedding\_Vectors[WordNet\_Words]$
$Query\_Vector \leftarrow Embedding\_Vectors[Query]$
$Context\_Terms \leftarrow$ Cosine($Query\_Vector, Embedding\_Vector$)
if IsPresent(Query_word,WordNet_Words) **then**
    $hyponyms \leftarrow$ WordNet.Hyponyms($Query$)
else
    $Similar2Query \leftarrow$ CosineSimilarity($Query_Vector, WorNet\_Vectors$)
    $hyponyms \leftarrow$ WordNet.Hyponyms($Similar2Query$)
end if
$Contextual\_Terms \leftarrow$ CosSimilarity($hyponyms, Term\_Vectors$)

Chapter 4

Experimentation and Results

4.1 Pre-Text

In this section, the rationale for the logic is tested against the ground truth contained in existing datasets. Using the observations in the previous section, the best set of parameters are applied to the Illinois data.

4.2 Implementation Tools

The experiments were performed in Python 3.7 on a Linux system provided by google colab environment. The tokenizing, Parts of speech tagging and chunking was performed using the NLTK library [Loper and Bird, 2002]. The colab environment provides us with a system following specifications 4.1.

GPU: 1xTesla K80, compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM
CPU: 1xsingle core hyperthreaded Xeon Processors @2.3Ghz, i.e.(1 core, 2 threads)
RAM: ∼12.6 GB Available
Disk: ∼33 GB Available

Table 4.1: Colab System Specifications

The implementation of the analysis of linguistic and statistic filter over annotated datasets are available here [1].

4.3 Comparison with existing Ground Truth

The term candidate recognition using noun phrases is verified on databases with existing ground truth. The literature by [Astrakhantsev, 2018][2] provides an open-source software written in scala for 13 state-of-the-art ATR methods. The results of these

[1]The code for implementation of statistical and linguistic filters available herehttps://colab.research.google.com/drive/16LXDT2eh3Ytsg-XDTBeD2qcRoEvxVKHl?usp=sharing

[2]Datasets can be downloaded from https://github.com/ispras/atr4s/releases/tag/v1.2

methods are compared with the list of 7 open datasets: Genia[Kim et al., 2003], FAO [Medelyan and Witten, 2008], Krapivin [Krapivin et al., 2009], Patents [Judea et al., 2014], ACL RD-TEC [QasemiZadeh and Handschuh, 2014], ACL RD-TEC 2.0 [QasemiZadeh and Schumann, 201 EuroParl [Koehn, 2005]. These datasets contain ground truth information by listing the terms present in them, as mentioned in Table:4.2.

Dataset	Domain	Docs	Words, thousands	Expected terms	Source of terms
ACL 2.0	Computational Linguistics	300	33	3059	Manual markup
Patents	Engineering	16	120	1595	Manual markup
GENIA	Bio-medicine	2000	494	35,104	Manual markup
Krapivin	Computer Science	2304	21,189	8766	Authors' keywords and Protodog glossary [Faralli and Navigli, 2013]
FAO	Agriculture	779	26,672	1555	Authors' keywords
ACL	Computational Linguistics	10,085	41,202	21,543	Manual annotation of 82000 candidates
Europarl	Politics	9672	63,279	15,904	Eurovoc thesaurus

Table 4.2: Dataset Statistics

Three of the datasets are manually annotated. Expected terms for ACL dataset was formed by manual filtering of top term candidates found by several simple ATR methods, so its recall is less than 100%, and some correlation with these simple ATR methods might be introduced. Two other datasets (Krapivin and FAO) were initially used for keyphrase extraction; their expected terms are a union of keyphrases of all documents, which is an approximation of actual terms. Texts and expected terms of Europarl dataset cover not only politics but multiple other (mostly related) domains.

4.3.1 Linguistic Filtering Pattern Analysis

Adding to the filters linguistic filters mentioned in the Section 3.2.5.1, two more filters; first, mentioned in literature [Astrakhantsev, 2018], and the second as shown as an example in NLTK tag manual are translated into NLTK chunk grammar and analysed for the results. The tag abbreviations are mentioned in 3.1

1. ATRS4[3]:
   $\{(<NN(S)?>|<JJ>|<NNP>|<NN(S?)>><IN>)*(<NN(S)?>)\}$
2. NLTK[4]:
   $\{<DT>?<JJ>*<NN*>\}$

[3]The tag pattern is translated to NLTK format from $(NN(S)?|JJ|NNP|NN(S?)IN)*(NN(S)?)$ as mentioned in the literature, It should be noted that there was no explanation describing the pattern. The translated pattern is an interpretation assuming the standard symbol meanings

[4]Grammar Pattern, as defined in NLTK literature. `http://www.nltk.org/howto/chunk.html`

3. Closed[5]:
   $\{<NN.*>+\}$
4. Open[6]:
   - $\{<NN.*>+<NN.*>\}$
   - $\{(<JJ>|<NN.*>)+(<NN.*>)\}$
   - $\{((<JJ>|<NN.*>)+|((<JJ>|<NN.*>)*(<NN.*><IN>)?)(<JJ>|<NN.*>)*)<NN.*>\}$
5. Personalized[7]:
   - $\{<NNP>1,3<NN?>?\}$
   - $\{(<JJ>|<NN>)<NN?>1,3\}$
   - $\{<DT>?<JJ>*<NN>\}$
   - $\{<DT>?<NN>+\}$

The results are tabulated in 4.3.

4.3.2 Statistical Filtering

All four statistical measures are applied over the terms generated using linguistic filters. A grid test is performed using the number of terms contained in the window of values. This window is defined starting from the maximum value up-to a percentage of the number of terms in the decreasing order. For example, in a set of 100 terms, 75% window will consist of the top 75 terms from the candidate set. The top {50, 65, 80, 95} percentage of terms are selected and analysed using the evaluation measures.

Each table below is the result for a single dataset. The maximum values are highlighted. The TF- IDF is calculated over each document, and later the value is normalized over all the documents by summing the values at each occurrence and dividing by the count of it. For C/NC measure the term candidate co-occurrences with context words are considered in its document. The recall is inversely proportional to the number of terms evaluated.

4.3.3 Evaluation Metrics

Evaluation, the performance of the system against ground truth is done using the below measures.

[5]As mentioned in [Dagan and Church, 1994]

[6]Translation of set of 3 patterns [Frantzi et al., 2000]:
$(Noun + Noun)$,
$((Adj|Noun) + Noun)$,
$(((Adj|Noun) + |((Adj|Noun) * (NounPrep)?)(Adj|Noun)*)Noun)$.
The NLTK pattern are ordered as per the length of chunked terms

[7]Empirically designed for Illinois Dataset refer Section 3.2.5.1

Datasets	Linguistic Filter	Precision	Recall	F-Measure
Acl2	**ATR4s**	0.218	0.328	0.262
	Open	0.405	0.510	0.452
	Closed	**0.492**	0.452	0.471
	NLTK	0.458	**0.644**	**0.535**
	Personalized	0.391	0.429	0.409
Patents	**ATR4s**	0.072	0.235	0.111
	Open	**0.255**	**0.715**	**0.376**
	Closed	0.245	0.520	0.333
	NLTK	0.229	0.692	0.344
	Personalized	0.209	0.566	0.305
Genia	**ATR4s**	0.191	0.211	0.200
	Open	0.264	0.325	0.291
	Closed	0.406	0.295	0.342
	NLTK	0.342	**0.426**	0.380
	Personalized	**0.430**	0.385	**0.406**
Krapivin	**ATR4s**	0.007	0.709	0.014
	Open	0.011	**0.636**	**0.021**
	Closed	**0.007**	0.497	0.013
	NLTK	0.009	0.719	0.017
	Personalized	0.010	0.666	0.019
FAO	**ATR4s**	0.001	0.738	0.003
	Open	0.001	0.895	0.002
	Closed	0.001	**0.950**	0.002
	NLTK	**0.002**	0.898	**0.004**
	Personalized	0.001	0.832	0.003
ACL	**ATR4s**	0.006	0.388	0.012
	Open	0.008	0.671	0.016
	Closed	0.009	0.712	0.018
	NLTK	0.012	0.542	0.024
	Personalized	**0.014**	**0.778**	**0.028**
Europarl	**ATR4s**	0.006	0.335	0.011
	Open	0.005	0.448	0.009
	Closed	0.007	0.524	0.013
	NLTK	**0.013**	0.385	**0.026**
	Personalized	0.012	**0.488**	0.023

Table 4.3: Linguistic Filters applied on Ground Truth Datasets

Statistical Filter	Top % Number of Terms	Precision	Recall	F-Score	AvP
C/NC- Value	**2**	**0.581**	0.011	0.021	**0.585**
	5	0.580	0.026	0.050	0.581
	10	0.559	0.051	0.093	0.567
	20	0.531	0.096	0.163	0.532
	40	0.489	0.178	0.261	0.485
	80	0.420	0.304	0.353	0.417
	95	0.432	0.368	**0.397**	0.428
	100	0.430	**0.385**	0.406	0.426
Residual - IDF	**2**	**0.811**	0.015	0.030	**0.816**
	5	0.736	0.034	0.066	0.726
	10	0.676	0.063	0.115	0.676
	20	0.603	0.112	0.188	0.600
	40	0.522	0.190	0.279	0.523
	80	0.453	0.324	0.378	0.448
	95	0.437	0.371	0.401	0.432
	100	0.430	**0.385**	**0.406**	0.426
TF- IDF	**2**	**0.811**	0.015	0.030	**0.816**
	5	0.736	0.034	0.066	0.726
	10	0.676	0.063	0.115	0.676
	20	0.603	0.112	0.188	0.600
	40	0.522	0.190	0.279	0.523
	80	0.453	0.324	0.378	0.448
	95	0.437	0.371	0.401	0.432
	100	0.430	**0.385**	**0.406**	0.426
C-Value	**2**	0.598	0.011	0.021	0.600
	5	**0.616**	0.028	0.053	**0.626**
	10	0.611	0.055	0.101	0.625
	20	0.554	0.099	0.168	0.554
	40	0.489	0.178	0.261	0.475
	80	0.426	0.308	0.357	0.423
	95	0.431	0.367	0.397	0.427
	100	0.430	**0.385**	**0.406**	0.426
ComboBasic	**2**	**0.638**	0.012	0.023	**0.651**
	5	0.629	0.029	0.055	0.650
	10	0.615	0.055	0.102	0.628
	20	0.600	0.107	0.182	0.601
	40	0.554	0.196	0.290	0.555
	80	0.459	0.330	0.384	0.446
	95	0.435	0.370	0.400	0.427
	100	0.430	**0.385**	**0.406**	0.426

Table 4.4: Performance of Statistical Filter over Genia Dataset

Statistical Filter	Top % Number of Terms	Precision	Recall	F-Score	AvP
C/NC- Value	**2**	0.164	0.003	0.007	0.226
	5	0.306	0.016	0.031	0.271
	10	0.370	0.040	0.072	0.364
	20	0.414	0.090	0.148	0.436
	40	0.423	0.187	0.259	0.426
	80	0.377	0.332	0.353	0.380
	95	0.387	0.403	0.395	0.390
	100	**0.391**	**0.429**	**0.409**	**0.396**
Residual - IDF	**2**	**0.649**	0.015	0.029	0.518
	5	0.584	0.033	0.062	**0.572**
	10	0.531	0.060	0.108	0.542
	20	0.483	0.109	0.178	0.466
	40	0.449	0.201	0.278	0.465
	80	0.406	0.358	0.380	0.415
	95	0.396	0.413	0.404	0.398
	100	0.391	**0.429**	**0.409**	0.396
TF- IDF	**2**	**0.649**	0.015	0.029	0.518
	5	0.584	0.033	0.062	**0.572**
	10	0.534	0.060	0.109	0.542
	20	0.483	0.109	0.178	0.466
	40	0.449	0.201	0.278	0.465
	80	0.406	0.358	0.380	0.415
	95	0.396	0.413	0.404	0.398
	100	0.391	**0.429**	**0.409**	0.396
C-Value	**2**	0.149	0.003	0.006	0.213
	5	0.312	0.016	0.031	0.312
	10	0.374	0.040	0.072	0.365
	20	**0.463**	0.101	0.166	**0.448**
	40	0.417	0.184	0.255	0.379
	80	0.385	0.338	0.360	0.383
	95	0.383	0.399	0.391	0.387
	100	0.391	**0.429**	**0.409**	0.396
ComboBasic	**2**	**0.708**	0.016	0.030	**0.779**
	5	0.545	0.029	0.056	0.602
	10	0.493	0.053	0.096	0.461
	20	0.480	0.104	0.171	0.461
	40	0.520	0.230	0.319	0.490
	80	0.422	0.371	0.395	0.406
	95	0.399	0.416	0.407	0.399
	100	0.391	**0.429**	**0.409**	0.396

Table 4.5: Performance of Statistical Measures over ACL2 Dataset

Statistical Filter	Top % Number of Terms	Precision	Recall	F-Score	AvP
Residual - IDF	**2**	**0.586**	0.033	0.062	**0.577**
	5	0.510	0.071	0.125	0.483
	10	0.482	0.134	0.210	0.496
	20	0.378	0.208	0.268	0.364
	40	0.295	0.324	**0.309**	0.291
	80	0.223	0.484	0.305	0.224
	95	0.213	0.549	0.307	0.211
	100	0.209	**0.566**	0.305	0.207
TF- IDF	**2**	**0.586**	0.033	0.062	**0.577**
	5	0.510	0.071	0.125	0.483
	10	0.482	0.134	0.210	0.496
	20	0.378	0.208	0.268	0.364
	40	0.295	0.324	**0.309**	0.291
	80	0.223	0.484	0.305	0.224
	95	0.213	0.549	0.307	0.211
	100	0.209	**0.566**	0.305	0.207
C-Value	**2**	**0.505**	0.028	0.054	**0.561**
	5	0.465	0.065	0.113	0.469
	10	0.437	0.121	0.190	0.433
	20	0.393	0.217	0.279	0.389
	40	0.314	0.345	**0.329**	0.320
	80	0.228	0.497	0.313	0.227
	95	0.208	0.536	0.300	0.208
	100	0.209	**0.566**	0.305	0.207
ComboBasic	**2**	**0.515**	0.029	0.055	**0.554**
	5	0.512	0.072	0.126	0.546
	10	0.464	0.129	0.202	0.448
	20	0.388	0.215	0.277	0.378
	40	0.319	0.350	**0.334**	0.319
	80	0.242	0.528	0.332	0.238
	95	0.215	0.553	0.309	0.210
	100	0.209	**0.566**	0.305	0.207

Table 4.6: Performance of Statistical Filter over Patents Dataset

- Precision: Percentage of candidate terms which were identified as correct terms.

$$Precision(P) = \frac{Number\ of\ RelevantTerms}{Number\ of\ Identified\ Terms} \tag{4.1}$$

- Recall: Percentage of ground truth terms which were identified as correct terms.

$$Recall(R) = \frac{Number\ of\ CorrectTerms}{Number\ of\ all\ Actual\ Terms} \tag{4.2}$$

- F1 Score: Harmonic mean of precision and recall. Only a trade-off between the previous two measures.

$$F1 - Score = \frac{2 * P * R}{P + R} \tag{4.3}$$

- Average Value Precision(AvP)[Prasath and Öztürk, 2011]: To evaluate 'ranking-quality' of statistical filters, the measure of Average Value Precision. This is adapted from information retrieval. More terms can be located at the beginning of the list more the value of average precision. Average Value Precision(AvP) in the task of ATR is calculated as follows.

$$AvP(k) = \frac{1}{k} \sum_{i} PrecTerm_i \tag{4.4}$$

  where,
  'k' the number of the ordered list of terms,
  'i' is the position of a term in the list of 'k'-terms
  '$PrecTerm_i$ is the precision calculated at the time of inclusion to the list of the i^{th}

Calculation of Recall in ATR is not complicated when the ground truth is present, whereas when it is not present, which is true in general, and gets complex. In [Milios et al., 2003], the author uses a 3 step approach. First, use a frequency threshold of 3 to get all the candidate terms. Second, count the number of real terms in this candidate term list. Third, use an additional C-value threshold of 0 to filter the term list obtained from step 2. Finally, calculate recall as the ratio of the number of terms after step2 to the number of terms after step 1.

It should be noted that as mentioned earlier, the state of the art performance in ATE is low. State of the art values for precision, in general, vary between 0.27 and 0.35, recall varies between 0.28 and 0.66, while F-measure (which decreases as the document length increases) falls between 0.27 and 0.45, as explained by [Hasan and Ng, 2014].

4.3.4 Conclusion from Testing over Term annotated datasets

It can be noticed that for ACL2, Patents and Genia; the performance is quite comparable. There is an increase in precision when the less number of terms are considered through

statistical filtering. Hence it can be implied that with more number of terms filtered using statical measures, the ATR performs better. Major factors that affect ATE performance measures are the length of documents, a lack of structural consistency (for instance, a lack of a defined index or abstract), topic changes, and the presence of uncorrelated topics in the same text. Moreover, it is also to be noted that the ground truth for the datasets is collected by manual marking which can have inconsistency from user bias. The remaining datasets do not perform poorly, having a precision of 1% and below. As mentioned in literature [Hasan and Ng, 2014], the performance decreases with the length of documents. During the grid test, we could formulate that the ComboBasic measure outperformed other measures in all the three datasets.

4.4 Evaluation of Embedding Models and Similarity Measures

Concluding from the state-of-the-art in word embeddings, We can infer that transformer model like BERT and ELMo are complex model, embeddings are suitable for specific tasks like disambiguation, sentiment analysis, topic modelling or question answering although the urge to use these complex models is resisted. In machine learning, the simplest solution tends to be a comparatively efficient one. Furthermore, since word vectors are needed only to identify similarities the less complex log-linear models are chosen. FastText is seen as an ideal choice since the model can handle out of word vocabulary. Along with FastText we train and test other algorithms on publicly available datasets. Testing flow is mentioned in algorithm 4. The implementation was tested on Google CoLab. [8]

Euclidean and Cosine distance can be a good measure for similarity as they help in measuring the closeness to the said vector. Nevertheless, in a higher-dimensional space cosine becomes more natural to calculate since it only uses the dot product of vectors in the calculation. To analyse results across models, we use the nearest neighbours of a word pointed by its vector using cosine distance to other word vectors.

4.4.1 Sample Text Corpus

4.4.1.1 The 20 Newsgroups Data-set

The 20 Newsgroups data set [Lang, 1995] is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. Since the dataset is publicly available, the corpus has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

The data is organized into 20 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other, while others are highly unrelated.

[8] `https://colab.research.google.com/drive/1qHnrcNBJHzF9jbBl7gnQq-YlPYxaw_4M`. The notebook contains the testing of the embedding algorithms on the mentioned datasets.

The 20 categories of 20 News Groups data-set.

comp.graphics	rec.autos	sci.crypt
comp.os.ms-windows.misc	rec.motorcycles	sci.electronics
comp.sys.ibm.pc.hardware	rec.sport.baseball	sci.med
comp.sys.mac.hardware	rec.sport.hockey	sci.space
comp.windows.x		
misc.forsale	talk.politics.misc	talk.religion.misc
	talk.politics.guns	alt.atheism
	talk.politics.mideast	soc.religion.christian

Table 4.7: List of the 20 newsgroups, partitioned according to the subject matter

4.4.1.2 Leipzig Wikipedia Corpora

The dataset [Goldhahn et al., 2012] contains 1 million randomly selected sentences in the English Wikipedia dumps. The texts are split into sentences. Non-sentences and foreign-language material have been removed. Also, word co-occurrence information which is useful for many applications; these data are precomputed and included as well.

4.4.2 Selection of Embedding Algorithm

To select a suitable word embedding algorithm for the implementation, multiple models are trained using; Word2Vec, NP-MSSG, GloVe and FastText. We skip testing ELMo and BERT models as these models are trained for more complex NLP tasks. All the mentioned algorithms are trained on the two datasets described above. Since the goal of the thesis is to find related words in a corpus, identical test case of 'word similarities' and 'word analogies' on each of the embedding algorithms is performed. Words similar to 'King' are queried using the nearest neighbours functions in all models, and top 5 results are tabulated. Simultaneously word analogies to [Man:Woman:: King:??] are also listed. The training accuracy is of little concern since interest lies in the word vectors from the embedding model, which is taken from the hidden layers of the training models. Finally, a suitable model is selected based on the findings.

4.4.3 Conclusion on Evaluation of Embedding Models with Cosine Similarity

Word2Vec a being a simpler model the word sense is not captured separately. Apple could mean the regular fruit or the company. However, the representations using Word2Vec is only one for both cases.

Algorithm 4 Implementation of the testing

The pseudo-code for the testing of available models in word embedding using the above-mentioned datasets.

```
import Gensim, NLTK, FastText                                  ▷ Initialize Libraries
for all algo ∈ Word2Vec, MSSG, Glove, FastText do
    for all dataset ∈ NewsGroups20, Wikipedia do
        data ← LOAD(dataset)                                   ▷ Sample Dataset
                                                               ▷ Preprocessing
        clean_data ← CLEAN(data)                               ▷ Remove special characters
        final_data ← TOKENIZE(clean_data)                      ▷ TreebankWordTokenizer
        model ← ALGO(final_data)                               ▷ training
        mostSimilar ← MODEL('word')                            ▷ nearest Neighbours
        WordAnalogies ← MODEL('word1' +' word2' −' word3')
    end for
end for                                                        ▷ Word analogies
```

The results from the tests performed by training of the embedding models on the datasets.

Model	**20 NewsGroup Dataset ('Word', similarity)**		**Leipzig Wikipedia Corpora ('Word', similarity)**	
	Nearest Neighbours of 'King'	Word Analogies ['man':'woman' :: 'King':???]	Nearest Neighbours of 'King'	Word Analogies ['man':'woman' :: 'King':???]
Word2Vec	'rodney', 42% 'solomon', 37% 'ajs', 35% 'brethren', 35% 'moses', 33%	'king', 60% 'woman', 40% 'rodney', 31% 'polishing', 27% 'beirut', 27%	'queen', 55% 'ruler', 54% 'harald', 53% 'prince', 52% 'kings', 51%	'king', 77% 'queen', 54% 'ruler', 44% 'woman', 44% 'princess', 44%
NP-MSSG	'rodney', 78% 'solomon', 69% 'testified', 67% 'gen', 66% '1937', 64%	'king', 67% 'rodney', 62% "o'", 53% 'testified', 53% 'tomb', 52%	'use–1', 19% 'recorded', 17% 'debut', 16% 'noticed', 15% 'darling', 15%	'queen', 81% 'princess', 53% 'elizabeth', 49% 'juliana', 44% 'woman', 44%
GloVe	'president', 90% 'presence', 89% 'messiah', 88% 'Creator', 88%	'king', 67% 'rodney', 61% 'banaian', 61% 'bahler', 59%	'zior', 87% 'burger', 77% 'charles', 76% 'edward', 76%	'king', 84% 'zior', 0.71% 'queen', 0.68% 'edward', 0.68% 'dedede', 0.67%
Fast Text	"hawking',", 75% "debunking',", 73% "king',",73% "fking',",72% 'faking',", 72%	'king', 75% 'woman', 54% 'kingcrane', 50% 'kin', 47% "king's", 47%	'king;', 84% 'king!"', 81% 'king,', 80% 'prince', 79% 'king)', 79%	'king', 75% 'queen', 57% 'woman', 50% 'noblewoman', 48%

Table 4.8: Top 5 Word Similarities and Analogies using multiple Embedding models

NP-MSSG performs word sense discrimination, meaning it gives different embedding per senses. Moreover, with the non-parametric model, the number of senses per word is also learnt by the model. Nevertheless the senses cannot be accessed intuitively as word vectors are differentiated with suffixes. Eg. Apple, Apple–1, Apple–2. Out of Vocabulary words do not have vector representations.

The GloVe model has a training time is lesser than word2vec, and also the results and previous benchmark [Pennington et al., 2014] shows it does better in semantic relatedness tasks compared to word2vec. The model has a significant memory footprint need. GloVe took 200 times more space than word2vec for the same dataset. - so certainly an issue for large corpus. Also, sense separation is absent just as in word2vec.

Since FastText is trained on n-grams that compose those words, Out of vocabulary words have fairly good vector representation, For example, the n-gram "av" is represented as a vector, and it learns its context from surrounding n-grams when it appears in behave, and in "have", "favourite". It can generate a representation of a vector that is close to the original despite some n-grams beings wrong (e.g. behave). This makes it useful in word sentence similarity tasks on a corpus with misspellings like tweets. One other advantage over word2vec is, again for the same reason of constructing word vectors from summing character n-gram vectors, is that word embeddings for even words that rarely occur in a corpus are of better quality. One main disadvantage like word2vec and glove, the model fails to disambiguate senses.

Also, from the tabulated results, we could conclude FastText model to be a better candidate as the analogies were closer to the expected word 'Queen', given not a huge corpus. Moreover, with lesser training time, a reasonably comparable model could be generated. Hence we proceed with our implementation using FastText.

Chapter 5

Implementation and Issues

5.1 Implementation on Illinois Dataset

In this section, the hypothesis is implemented on the Illinois Case decisions data to extract terms. The logic is implemented using python using the system specification mentioned in 4.1. The process mentioned in the methodology is followed with the top parameters decided from the results of the experiment on existing ground truth. The implementation ATR on Illinois Dataset can be found in the Jupyter notebook hosted by Google Colab [1].

5.1.1 Data Collection

The text used to for the process is collected from US court-decisions found in The Illinois Bulk dataset which is made publicly available by the Case-Law Access Project(CAP). The data was bundled by digitizing roughly 40 million pages of court decisions. Roughly 40000 bound volumes owned by Harward Library were scanned, and optical character recognition (OCR) was used to extract the text of every case and then combined with meta-data created for each volume, including attributes like unique barcode, reporter name, title, jurisdiction, publication date. The dataset is available in the JSON format and has close to 0.2 million court decisions (CD's). Each CD has information along the XML tree separated by their respective tags. Each CD has one or more arguments by judges separated by 'opinion' tags. Since the scope of research relates to finding related terminologies, only the textual content available in the 'text' tag is extracted for processing ignoring the rest. The detailed statistics are mentioned in Table 5.1

Case Decisions	183,146
Opinions	194,366
Sentences	14,187,674
Tokens	337,142,906
Unique Tokens Case Differentiated	837912
Unique Tokens Case Undifferentiated	761102

[1]The code for implementation of statistical and linguistic filters available here`https://colab.research.google.com/drive/1eBtXNAd7-tP4muqIpTBbcWOaZi5CHidJ?usp=sharing`

Table 5.1: CAP Dataset Statistics

5.1.1.1 Memory Issue

The bulk dataset is downloadable as 'xzip' compressed file. The file extracts into a text file where each line is a JSON object. The dataset decompresses to approximately 2GB on the memory. Also extracting text and tagging POS expands the memory considerably, eventually maxing the system out. To avoid excessive consumption of the memory, the file is streamed each line at a time to process one JSON object at a time. This is accomplished using a python generator objects that yields each object at a time, which is processed to generate terms and later the variables are cleared to avoid stacking up space. Details about python generator is mentioned in Appendix A 6.1.

5.1.2 Tokenizing text

The text is first tokenized into sentences using the PunktSentenceTokenizer. Then each sentence is tokenized into words using a whitespace tokenizer. There are multiple ways to tokenize text in NLTK. Few of them are mentioned with the example following the section. For the downstream task of POS tagging, the tokenizer is selected in such a way that the tagger makes accurate POS tagging of the tokens. Hence the text is tokenized using white space. This tokenizer keeps the contractions together, which is essential to not clutter the tagging for the chunking task.

E.g. "In Düsseldorf I took my hat off. But I can't put it back on."

- **TreebankWordTokenizer**

  ['In', 'Düsseldorf', 'I', 'took', 'my', 'hat', 'off', '.']

  ['But', 'I', 'ca', "n't", 'put', 'it', 'back', 'on', '.']

- **WordPunctTokenizer**

  ['In', 'Düsseldorf', 'I', 'took', 'my', 'hat', 'off', '.']

  ['But', 'I', 'can', "'", 't', 'put', 'it', 'back', 'on', '.']

- **WhitespaceTokenizer**

  ['In', 'Düsseldorf', 'I', 'took', 'my', 'hat', 'off.']

  ['But', 'I', "can't", 'put', 'it', 'back', 'on.']

- **RegexpTokenizer**

  By Setting Regular expression to "[\w']+"

  ['In', 'Düsseldorf', 'I', 'took', 'my', 'hat', 'off']

  ['But', 'I', "can't", 'put', 'it', 'back', 'on']

- **TweetTokenizer**

  ['In', 'Düsseldorf', 'I', 'took', 'my', 'hat', 'off', '.']

  ['But', 'I', "can't", 'put', 'it', 'back', 'on', '.']

5.1.2.1 Tokenizer Issues

Multiple tokenizers like the above were evaluated for, e.g., the default NLTK 'WordPunctTokenizer' split the contractions into two separate words(like 'couple's' to 'couple' and 's'). Although the contractions are hardly found in terminology, they can add to the false positives of the algorithm. Hence the words are tokenized at white space.

5.1.3 Parts of Speech Tagging

The tagger used here is a pre-trained model from NLTK distribution called 'Averaged Perceptron Tagger'. It has a 97.1% evaluation on 130000 words of text from the wall street journal. The POS tagger has better accuracy when the morphology of text is unmodified. Also, the parts of speech rely on long-term dependencies. Hence a minimalistic test preprocessing is performed before tagging the corpus.
The POS tags and their description is detailed in Table 3.1 in the methodology section.

5.1.3.1 Tagging Issues

The Illinois dataset contains close to 180 thousand documents. To get the complete statistics of the dataset, the system needs to have a memory higher than 12GB as the system memory maxes out if a tagging is performed on the complete dataset as a whole. Hence the tokenizing, tagging and chunking is performed on each case decision, and only the chunks or noun phrases are saved in the memory for future use. POS tag statistics for 10K case decisions available in Table 5.2.

Tag	Count	Tag	Count
NN	4362612	**WDT**	125472
IN	3287295	**WRB**	75484
DT	2909454	**WP**	41620
NNP	1945092	**EX**	36659
JJ	1349268	**RP**	26761
CD	1127266	**JJR**	22295
VBD	1093012	**JJS**	15014
NNS	936362	**NNPS**	13857
VB	771519	**RBR**	13764
RB	761625	**PDT**	13335
VBN	733589	**FW**	11571
CC	719196	**$**	3662
TO	681956	**RBS**	3647
PRP	528901	**WP$**	2477
VBZ	453524	**UH**	510
VBG	372890	**POS**	321
PRP$	257296	**LS**	251
MD	238522	**SYM**	14
VBP	233253	**”**	2

Table 5.2: Count of Tags in the First 10K Case Decision

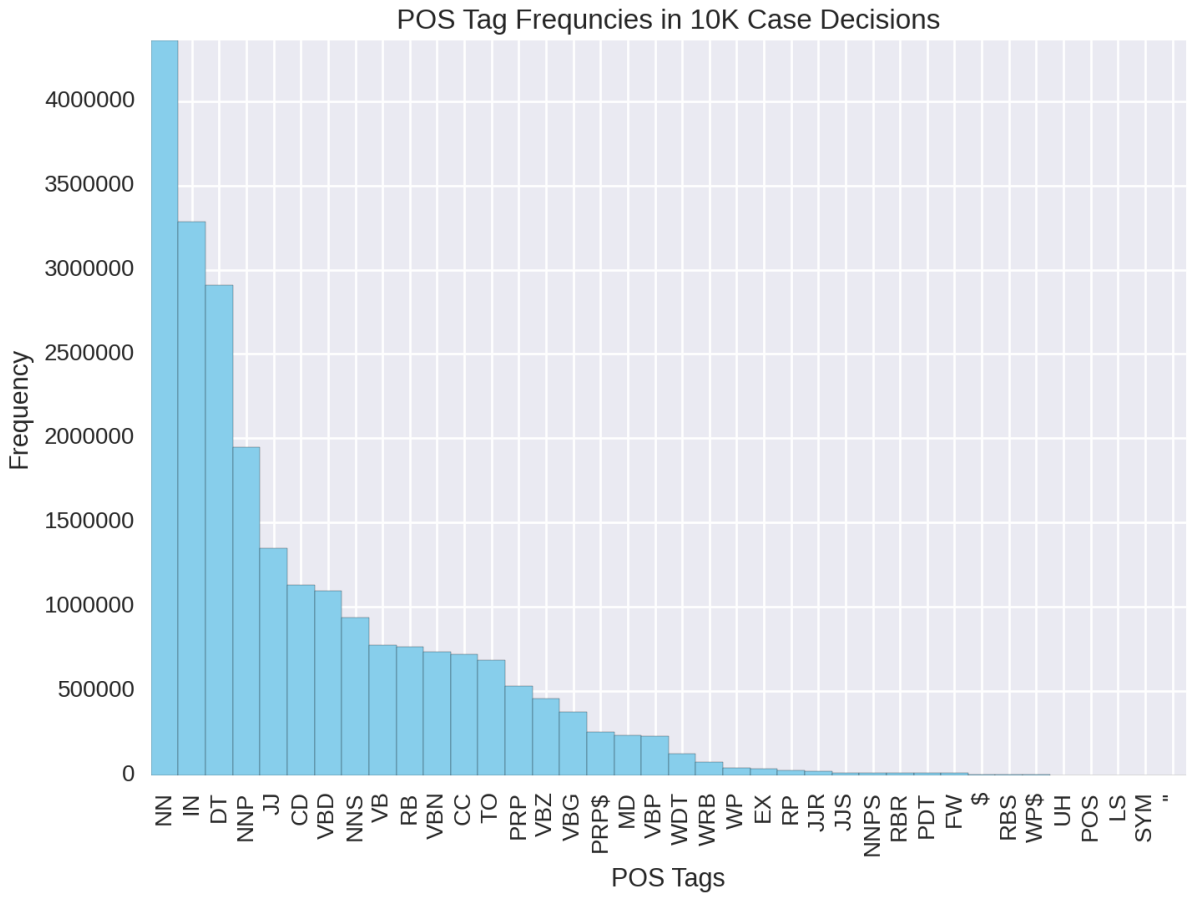

Figure 5.1: Graph of POS Tag Frequencies

5.1.4 Linguistic Filtering using Noun Phrase Chunking

The Noun Phrase detection is performed using the NLTK library. Finding the right tag pattern is a crucial part of the experiment. Different domains can have different patterns as per the terminologies available in them. There is no easy way to determine a tag pattern. In this implementation, a tag pattern is decided empirically. Various tag patterns were analysed as will be mentioned in the next section and then finally arrived with a group of 3 patterns. It is to be noted that the library doesn't take into account the nested pattern. Hence we declare the patterns in the order of chunking the longest terms first. Once the terms with the longest length of words are chunked, those are discarded from further nested chunking although it is to be noted that split/merge methods in NLTK can be used to split existing chunks.

5.1.4.1 Chunk Grammar Pattern Analysis on Illinois Dataset

Analysis of POS patterns mentioned in different ATR literature applied on to the subset (1000 CD's) of the Illinois dataset.

- ATRS4[Astrakhantsev, 2018]:
  $\{(< NN(S)? > | < JJ > | < NNP > | < NN(S?) >< IN >) * (< NN(S)? >)\}$
  2245 variations of tag patterns were found and the top/bottom 5 variations are mentined in Table 5.3

Pattern Possibilities	**Example Term**	**Counts**
NN	opinion	202913
NNS	indicates	45876
JJ NN	aggravated battery	41327
NN NN	trial court	24445
NN IN NN	appeal by defendant	12581
...	...	...
NNP NNP NN NN NN NNS IN NN	Cook County circuit court judge sits in judgment	1
NNS IN NN IN NNS NN	causes of action for premises liability	1
JJ NN IN NN IN NN IN NNS IN NN	court's consideration in lieu of live of fers ...	1
NN IN NN NN NN NNS IN NNP NNP NN	denial of plaintiff's duty disability benefits...	1
NN NNP NNP JJ NN NNS	addition Dr Snyder's medical malpractice costs	1

Table 5.3: Sample Terms of ATR Tool Linguistic Filter [Astrakhantsev, 2018]

- Closed Filter Chunk Grammar [Dagan and Church, 1994]
  $\{< NN.* > + < NN.* >\}$

577 patterns were of unique noun sequences were found, snapshot available in Table 5.4.

Pattern Possibilities	**Example Term**	**Counts**
NN	opinion	300639
NNS	reports	71093
NNP	Children	70275
NN NN	et seq	34530
NNP NNP	JUSTICE HARTMAN	22401
...	...	...
NNP NNP NNP NNP NNP NNP NNP NNP NNP NN	TO REENGAGE THE OPERATOR PULL RED HANDLE TOWAR...	1
NN NNP NNP NNPS NNS	court Plaintiff Saco Industries appeals	1
NN NNS NN NNS NNS NNS NNS	tax returns cash receipts journals sales invoices	1
NNP NNP NNP NNP NNP NNP NNP NN	A Miller M Kane Federal Practice Procedure §	1
NNP NNP NNP NNP NNS NNS	Area Five Violent Crimes police headquarters	1

Table 5.4: Sample Terms of "Closed" Linguistic Filter

- Open Filter Chunk Grammar[Justeson and Katz, 1995][Frantzi et al., 2000]
  - $\{((< JJ > | < NN >) + |((< JJ > | < NN >) * (< NP >< IN >)?)(< JJ > | < NN >)*) < NN >\}$
  - $\{(< JJ > | < NN >) + (< NN >)\}$
  - $\{< NN > + < NN >\}$

  721 variations of tag patterns were found and the top/bottom 5 variations are mentioned in Table 5.5

- Grammar, as defined in NLTK literature.
  $\{< DT >? < JJ > * < NN.* >\}$

  30 variations of tag patterns were found, and the top/bottom five variations are mentioned in Table 5.6

- Personalized Chunk Grammar Empirically designed filter based on identifying probable noun phrases in the Illinois Text. The final list of 13 ATR tag patterns used is described in the Table5.7.
  - $\{< NNP > \{1, 3\} < NN? >?\}$

Pattern Possibilities	**Example Term**	**Counts**
NN	opinion	225283
NNP	Ill	76371
JJ NN	electrical contractor	61924
NNS	defendants	51727
NN NN	malpractice action	43121
...	...	...
JJ NNS JJ NNPS	Mexican fagots sic Torres	1
NNP NNP NN NN NNP NNP NNP NNP	Cook County circuit court's Forensic Clinical ...	1
NN NN NNP NN NN NN NNP	fraud count II negligent misrepresentation cou...	1
NNS NNP NNPS NNP NNP NNP NNP	defendants Eljer Industries Inc Eljer Manufact...	1
NN NNP NN NNP NNP NNP NNP NNP NNP NNP	park Lewis v Jasper County Community Unit Scho...	1

Table 5.5: Sample Terms of "Open" Linguistic Filter

Pattern Possibilities	**Example Term**	**Counts**
NNP	JUSTICE	276694
NN	action	274790
DT NN	the opinion	232765
NNS	sections	74062
...	...	...
JJ NN	minor child	43794
DT JJ JJ JJ JJ NN	a real petite little white girl	3
DT JJ JJ JJ NNP	a 35-page civil federal RICO	3
JJ JJ JJ NNP	defendant North American Van	3
JJ JJ JJ JJ NN	attempted aggravated criminal sexual assault	2
JJ JJ NNPS	cross-wiring destroyed Claimants	1

Table 5.6: Sample Terms NLTK Linguistic Filter

– $\{(< JJ > | < NN >) < NN? > \{1, 3\}\}$
– $\{< DT >? < JJ > * < NN >\}$
– $\{< DT >? < NN > +\}$

Pattern Possibilities	**Example Term**	**Counts**
DT NN	the opinion	161953
NN	violence	118393
NNP	State	91426
JJ NN	aggravated battery	66843
NN NN	drinking pop	34415
NNP NNP	JUSTICE CAHILL	29572
NNP NN	Wade gathering	13772
NNP NNP NNP	GD GD GD	13416
JJ NN NN	first degree murder	7002
NN NN NN	jury trial defendant	4115
NNP NNP NN	Barbara Anderson codefendant	2509
JJ NN NN NN	criminal intent flight association	685
NNP NNP NNP NN	Illinois Supreme Court's opinion	631
NN NN NN NN	law enforcement bis ability	380

Table 5.7: Sample Terms of "Personalised" Linguistic Filter

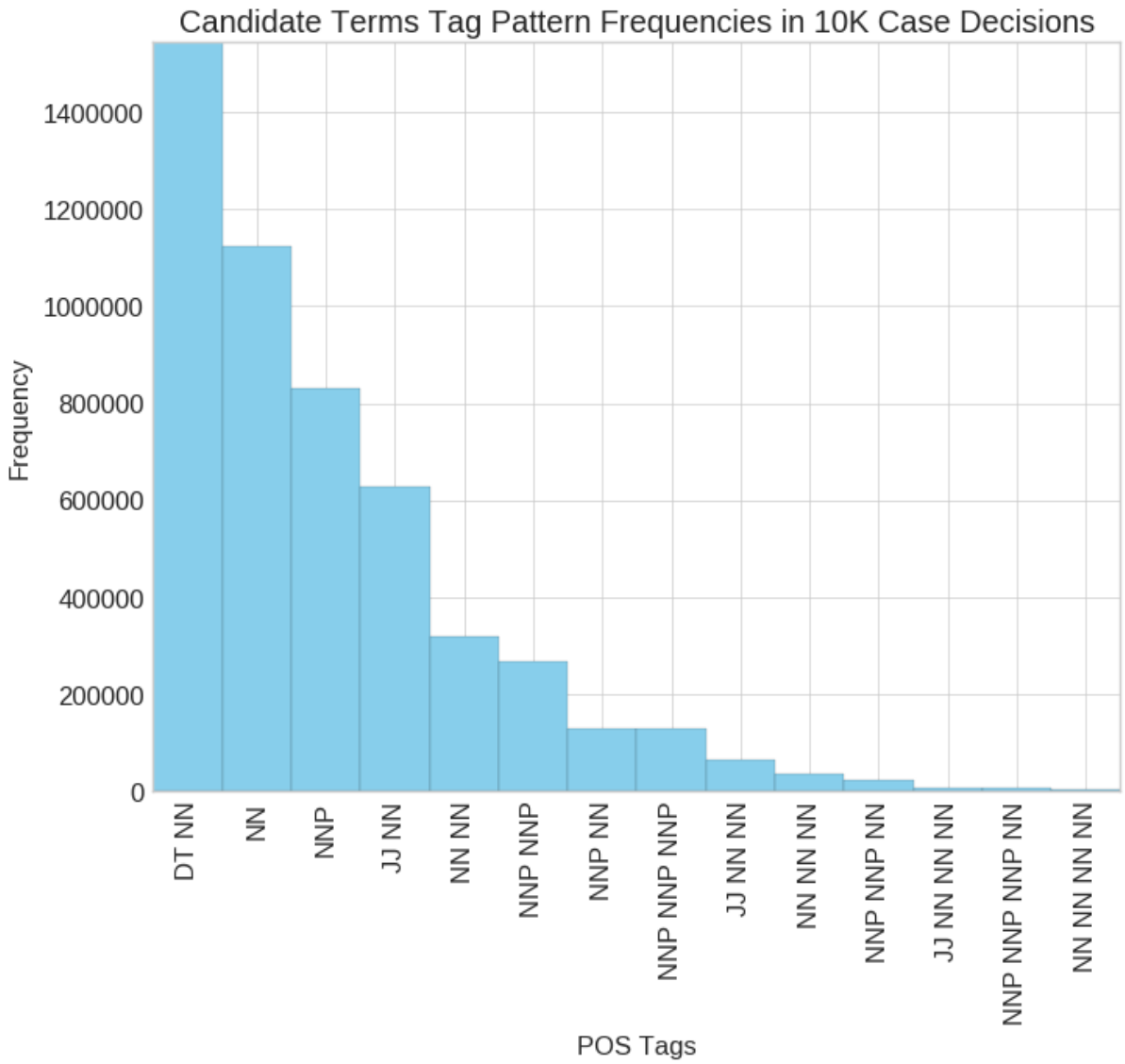

Figure 5.2: Graph of Candidate Terms Tag Pattern Frequencies

5.1.4.2 Linguistic Filtering Issues

From the above analysis, it is visually evident that personalized filters are more effective than the 'Open' or 'Closed'. In table 5.3 5.45.5 long sequences of noun phrases are recognized which are incorrect in constituting as terms. Also, the Close Filter recognized all the common nouns (like 'children', 'opinion')available in the text, which increases the recall considerable.

Whereas in the personalized filter 5.7, terms like 'First Degree Murder' which is a popular term used in the legal domain were recognized. Occasionally false positives like 'GD GD GD' get recognize. Such occurrences get filtered in the statistical process due to its rare frequency.

Another main issue remains for the accuracy with POS tagging, as it can be noticed in some cases, the Parts of speech was recognized incorrectly. E.g. in Table 5.7 'drinking'

was recognized as a noun, although its correct parts of speech is a 'verb'. This increases the false positives in the final result.

5.1.5 Statistical Filtering of Candidate Term

Although there is no existing ground truth for the dataset to confirm the hypothesis, the statistical measure can rank the terms in order to decrease the number of possible non-terms that can be sifted across by an expert or a system. Tables 5.8, 5.9, 5.10, 5.11, 5.12 give the top/bottom terms as per the measures. The terms were extracted from 1000 documents from the Illinois dataset.

5.1.5.1 Issues with Statistical Measures

In Table TF-IDF 5.8 and R-IDF 5.9, most of the single-word terms are ranked higher. This is possible as the measure is dependent heavily on the frequency of the term. Single-word terms have more frequency of occurrence than the multi-word noun phrase. Hence we can see more of document relevant nouns are listed in the topmost terms. Also, the top/bottom terms in r-idf are almost similar; hence it is conclusive that they are similar. This is perceivable since R-IDF factors the TF-IDF.

Whereas C-Value 5.10, C/NC- Value 5.11 and Combo-Basic 5.12 give better results in recognizing terms

5.2 Semantic Information Extraction

The task of finding terms based on its semantic relatedness to the query term is achieved using the below process. First, all the terms and the query are featurised by converting them into points in an n-dimensional vector space. Next, the cosine distance between the vector and all the terms is calculated. Finally, the terms are sorted in the increasing order of the distance between the query and other terms. The relatedness to the query is maximum with the first term in the list of ordered terms and decreases with each term lower in the order.

5.2.1 Feature Extraction of Terms

The terms are converted into vectors by extracting the corresponding embeddings from the trained word2vec model. Since the terms also consist of multi-words, the vector for a term can be approximated in two ways. First by summing up vectorised character n-grams that make up the terms as in case of fastText model out of vocabulary feature. Second by summing up vectors of the n-grams that make up the term. Details are mentioned in the previous chapter. The word2vec embedding model can either be trained on the dataset or another dataset as the syntactic and semantic feature is a generalized property of any language. Each element in the word vector[2] approximates a piece of information relating to the database hence constituting feature of the word.

[2]Word vector and word embeddings mean the same and are interchangeable used

Top Terms	Value
Roxanne	0.67
Ampad	0.63
Battalion Chief Bonnar	0.56
Shannon	0.53
S.S	0.51
Newsom	0.49
AFM	0.49
Herbolsheimer	0.48
Vanzandt	0.47
Conrail	0.47
Gunthorps	0.47
Horney	0.47
Carus	0.47
Lumber	0.46
Forsythe	0.46
SSMG	0.46
Mongolis	0.46
MFS	0.45
Bedar	0.44
DeLaire	0.43
Kirsten	0.43
strap	0.42
Audi	0.42
Metropolitan	0.42
Heidenhain	0.41
Alivio	0.41
BABCO	0.41
Country Mutual	0.41
Waibel	0.41
Rosalind	0.39

Bottom Terms	Value
slow student	0.00
Basically	0.00
defendant police	0.00
Kankakee police	0.00
JUROR TERLEP	0.00
light blue	0.00
Bascom	0.00
Zoot's opinion	0.00
Irene Keagle Similarly	0.00
desk clerk	0.00
wooded area east	0.00
Fancher's testimony	0.00
Theresa Billingsley	0.00
flight risk	0.00
Boose court	0.00
Mercy	0.00
Corey Gerlach.' McNeal	0.00
right mind	0.00
adequate voir dire	0.00
juror see	0.00
Dillon's	0.00
Christopher's inhaler	0.00
precise estimation	0.00
attempted revenge	0.00
outweigh evidence	0.00
Berglund Francis Wood	0.00
Jury Selection Defendant	0.00
probable cause probability	0.00
defendant's core	0.00
defendant's twin	0.00

Table 5.8: Top and Bottom 30 terms in TF-IDF

Top Terms	Value
Roxanne	0.21
Ampad	0.17
Battalion Chief Bonnar	0.10
Shannon	0.07
S.S	0.05
Newsom	0.03
AFM	0.03
Herbolsheimer	0.02
Vanzandt	0.01
Conrail	0.01
Gunthorps	0.01
Horney	0.01
Carus	0.01
Lumber	0.01
Forsythe	0.00
SSMG	0.00
Mongolis	0.00
MFS	-0.01
Bedar	-0.02
DeLaire	-0.03
Kirsten	-0.03
strap	-0.03
Audi	-0.04
Metropolitan	-0.04
Heidenhain	-0.04
Alivio	-0.04
BABCO	-0.05
Country Mutual	-0.05
Waibel	-0.05
Rosalind	-0.07

Bottom Terms	Value
slow student	-0.46
Basically	-0.46
defendant police	-0.46
Kankakee police	-0.46
JUROR TERLEP	-0.46
light blue	-0.46
Bascom	-0.46
Zoot's opinion	-0.46
Irene Keagle Similarly	-0.46
desk clerk	-0.46
wooded area east	-0.46
Fancher's testimony	-0.46
Theresa Billingsley	-0.46
flight risk	-0.46
Boose court	-0.46
Mercy	-0.46
Corey Gerlach.' McNeal	-0.46
right mind	-0.46
adequate voir dire	-0.46
juror see	-0.46
Dillon's	-0.46
Christopher's inhaler	-0.46
precise estimation	-0.46
attempted revenge	-0.46
outweigh evidence	-0.46
Berglund Francis Wood	-0.46
Jury Selection Defendant	-0.46
probable cause probability	-0.46
defendant's core	-0.46
defendant's twin	-0.46

Table 5.9: Top and Bottom 30 terms in Residual - IDF

Top Terms	Value
Ill App	11,459.59
trial court	8,131.50
circuit court	2,968.68
defendant	1,721.38
L Ed	1,629.13
summary judgment	1,454.62
court	1,450.19
Ill	1,421.81
N.E.2d	1,290.20
appellate court	1,069.22
supreme court	1,052.31
Supreme Court Rule	866.73
trial judge	784.10
evidence	772.15
case	749.95
plaintiff	702.53
section	693.46
ILCS	643.61
instant case	573.68
Ill Rev	563.61
present case	547.95
new trial	529.28
State	522.75
defense counsel	499.13
reasonable doubt	498.13
time	477.14
motion	465.72
et seq	464.55
due process	459.00
defendant's motion	435.19

Bottom Terms	Value
auction sale	-14.99
valid driver's	-14.99
degenerative disk	-14.99
guardian ad	-16.06
error doctrine	-16.06
DEFENDANT MAXWELL Yes	-16.32
Recreational Use	-17.13
Edison Co	-17.13
special town	-17.13
present cash	-17.13
amendment right	-17.39
money security interest	-17.95
court's grant	-19.80
Olson Horek	-22.48
Vanek Olson	-22.48
rule unit	-23.55
good conduct	-24.62
visitation interference statute	-26.12
loss doctrine	-26.76
State County	-35.32
ad litem	-40.14
Ed	-41.91
App	-43.57
Illinois Human	-50.31
seq	-50.74
Juvenile Court	-76.00
Ill Ct	-94.19
Adm Code	-138.08
Court Rule	-282.58
	-4,933.06

Table 5.10: Top and Bottom 30 terms in C-Value

Top Terms	Bottom Terms
asset value	impeachment method
purported subrogee	circuit court's stay
possession regardless	compensation claim
People v Stingley	support enforcement
agreed-upon sentence	review board
Three Corporations	T&S Signs
Puccia	Melrose Park
department paraphernalia	court's denial
Leach court	W LaFave
Holdridge J	second degree
sexual offenders' program'	Illinois Central
Marital Balance Sheet	v Virginia
Trapkus	X offender
pretrial memorandum petitioner	Cl
mental retardation category	judge's ruling
childlike level	College District
T]his sentence	valid driver's
detailed time	degenerative disk
specific disposition	present cash
Sage's murder	amendment right
North Loop Guidelines	money security interest
common denominator	court's grant
loss policy	Ill Ct
State agree	Olson Horek
Minnesota Mutual	good conduct
defendant's incarceration subject	loss doctrine
Bill Kruse d/b/a	App
I]f someone	Court Rule
Germania Federal Savings	Adm Code
Dr Marx' report	Adm Code

Table 5.11: Top and Bottom 30 terms in C/NC - Value

Top Terms	Value
v	2,475.90
defendant	727.19
defendant's	677.23
court	590.52
Illinois	493.24
Dr	472.21
trial	447.97
Inc	441.01
People	431.32
Co	429.39
People v	421.69
case	382.11
court's	357.16
plaintiff	329.54
pursuant	327.74
County	309.38
Act	308.63
testimony	306.68
claim	290.82
cannot	286.01
motion	276.63
plaintiff's	271.09
order	262.97
action	262.11
law	250.67
State	238.49
defense	236.94
Insurance	221.65
evidence	221.63
argument	219.05

Bottom Terms	Value
trustees'	0.00
out'	0.00
Vehmeyer	0.00
E]vidence	0.00
researcher	0.00
fracas	0.00
Holcomb's	0.00
self-employed businessman	0.00
epithet	0.00
Monka	0.00
Spray	0.00
offense(s	0.00
Raman	0.00
chordee	0.00
intrafamilial warfare	0.00
Practice's	0.00
LRRM	0.00
step-grandfather	0.00
complete severing	0.00
nearby bush	0.00
Tony's	0.00
favored suitor	0.00
Covered	0.00
former paralegal	0.00
Renaldo	0.00
Klotz	0.00
tension	0.00
Chaplinsky	0.00
cxiv	0.00
submissiqn	0.00

Table 5.12: Top and Bottom 30 terms in ComboBasic

5.2.1.1 Word2Vector Training using FastText

The word2vector model is generated by training the fastText on Illinois dataset. Only text from the dataset is extracted and concatenated. This concatenated text is used as input for training the embeddings using fastText. As mentioned in the earlier chapter, skip-gram has shown to give better semantic results than CBOW. The skip-gram model is trained using default hyperparameters with 5 epochs, context window of 5 and for unigrams. We can see in the results section that within 5 epochs the model has generalised and approximated comparable semantic features in the dataset.

5.2.1.2 Word2Vector Using Pre-Trained Wikipedia Embeddings

In the previous subsection, we trained fasttext model using the CD from Illinois Dataset. The word vectors also can be derived from a model that is trained on another dataset Wikipedia. Pre-Trained Wikipedia embeddings can be used to generate embeddings for the extracted using ATR [3]. The embeddings used have 300-dimensional vectors. Examples for related vectors are displayed in the Table5.14 and Table 5.16.

5.2.1.3 Issues with Featurizing Terms

An embedding model of the size in relation to a machine is huge if it occupies most of the memory in a system, or prohibits any more operations from being performed leading in eventually crashing the system. The pre-trained Wikipedia-300 dimensional model used in the implementation occupies a space of more than 9 gigabytes. Such consumption is huge if the underlying available machine space is 12 gigabytes. As more space would be required to collect vectors from the terminologies.

There are multiple ways to reduce the consumption of memory. First, it should be noted that in both trained and pre-trained, the embeddings are only needed for featurizing the terminologies to find semantic relations within the terminologies and not between the embeddings. Hence one way to save the memory is to clear the embedding model once we have all the vectors for the terminologies extracted. Another way to reduce is by reducing the pre-trained model to a lower dimension like 100. For an embedding algorithm trained with a comparable learning rate and epochs, generalises a word vector with 100-300 dimension. Hence a pre-trained model can be reduced to a lower dimension to fit the same purpose without much loss in performance. More details can be found in Appendix A 6.1

5.2.2 Similarity Querying

As discussed earlier, there are multiple methods to calculate similarity. However, In a multi-dimensional vector space cosine efficiently and accurately maps to its similar entity. The terms vectors are concatenated into a matrix, and using matrix properties, the cosine of the vector against all the terms extracted are measured to give top related terms.

[3]The embeddings made available publicly by Facebook in the following URLhttps://dl.fbaipublicfiles.com/fasttext/vectors-wiki/wiki.en.zip

In the Table5.13, it can be noticed that similarity is also found against the multi-word terms using embeddings which is not possible when an embedding is trained on uni-grams tokens.

Related Terms of "Narcotics"	Similarity Value
U S Narcotics	0.96
Narcotics Team	0.96
Narcotics	0.96
Narcotics Act	0.95
Narcotics Unit	0.95
Narcotics Abuse	0.94
Narcotics Unit Vice	0.94
Narcotics Task Force	0.94
Narcotics Agents	0.94
Narcotics Detail	0.94
Narcotics Drug Act	0.93
Narcotics Bureau	0.93
Narcotic Unit	0.93
Narcotic Drugs	0.93
Narcotics Office	0.93
Narcotic act	0.93
Narcotic '	0.93
Narcotic Drug	0.93
Narcotic Drugs	0.92
Narcotics Branch	0.92
New York Narcotics	0.92
Non-Narcotics	0.92
Narcotics Agent Larry	0.92
Narcotics Cases	0.92
Narcotic Drags	0.92
Narcotic Drugs Act	0.92
Narcotic Act	0.91
Narcotics Division	0.91
Narcotics Court	0.91

Related Terms of "Weapons"	Similarity Value
Weapons	0.98
Weapons Ban	0.97
Weapons Act	0.96
Weapons Case	0.94
Weapons Class	0.93
Deadly Weapons	0.93
Deadly Weapons act	0.92
Weapon	0.92
Deadly Weapons Act	0.92
Weapon	0.92
Weapon '	0.91
Weapon K.H	0.91
Weapons Statute	0.87
Weapons Firing Range	0.87
Weapon Beyond	0.87
Deadly Weapon Act	0.87
Weapons statute	0.87
Deadly Weapon act	0.87
Deadly Weapon	0.86
Special Weapons Assault	0.86
Offense Charged Unlawful sale	0.85
Weapons Evidence	0.84
Weapons provision	0.84
Weapons Dealer Control	0.84
Unlawful Use	0.83
Unlawful Entry	0.83
Misdemeanor Unlawful Use	0.83
Unlawful act	0.83
Unlawful Ouster	0.83

Table 5.13: Top 30 similar Values for "Narcotics" and "Weapons".

5.2.3 Contextual Querying

Given a subject or context, the terminologies relating to it could be recognized. This can be achieved using an external knowledge source. In this literature, the information is derived using WordNet lexical resource. The information relating to the terms of a subject is extracted from WordNet as a vector. This vector is further used to find the related terms among the extracted terms.

5.2.3.1 WordNet

The WordNet dataset is structured as a graph. The words are grouped into sets of cognitive synonyms (synsets), each expressing distinct concepts. Synsets are interlinked using conceptual-semantic and lexical relations. Words have encoded relation among

Related Terms of "Narcotics"	Similarity	Related Terms of "Weapons"	Similarity
U S Narcotics	0.904	Weapons'	0.872
Non-Narcotics	0.900	Weapons Ban	0.794
Narcotic	0.838	the weaponry	0.787
Narcotics Act	0.836	weapons.'	0.785
narcotics'	0.831	Weapons Case	0.781
Federal Narcotics	0.830	weapons'	0.779
narcotics.'	0.830	unusual weapons.'	0.776
Narcotics Section	0.828	Weapons Class	0.774
SNIP Unit Narcotics	0.826	all weaponry	0.773
Narcotics Unit	0.824	other weaponry	0.766
narcotics cure	0.821	a weapon,'	0.764
Narcotics Team	0.821	Weapons Rev	0.760
Narcotics Branch	0.820	certain weaponry	0.750
narcotics case	0.819	military weaponry	0.747
Narcotics Agents	0.816	concealed weapons'	0.747
narcotics.' v	0.815	the weapon.was	0.745
Narcotics Abuse	0.813	no weaponry	0.745
nonnarcotics case	0.810	a weapon'	0.744
Narcotics Court	0.807	a weapon	0.743
narcotics drug	0.804	the weapon.'	0.736
Narcotics Bureau	0.803	particular weapon(s	0.731
Narcotics United	0.803	seized weapon	0.730
Narcotics Drug Act	0.801	either weapon	0.730
New York Narcotics	0.800	Weapon	0.730
narcotics arrest	0.799	prior weapon	0.729
narcotics sale	0.799	small weapon	0.729
a narcotic	0.798	likely weapon	0.729
non-narcotic drug	0.796	small-caliber weapon	0.728
Narcotics Unit John	0.795	deadljr weapon	0.727
narcotics court	0.795	This weapon	0.727
narcotics offense	0.789	ous weapon	0.725

Table 5.14: Top 30 similar Values for "Narcotics" and "Weapons" using Pre-Trained Wiki-Embeddings

other synsets as it super-subordinate relation (also called hyperonymy, hyponymy or ISA relation). E.g. 'bed', 'chair' are hyponyms of 'furniture'. Hence here we consider the inherited topic incase of 'bed' or 'chair' to be 'furniture'. The process of finding terms relating to a subject is found in four steps.

- First given a query term 'furniture', its hypernyms are extracted from WordNet.
- Second, the word vectors for the hypernyms are looked up in the word2Vec model; if they are absent in the vocabulary of the model, then they are created using character n-grams.
- Next, all the word vectors are combined by summing them up.
- Finally, the summed up vector is used to find the nearest terms in the ATR list using cosine distance.

In instances the query word not available in the WordNet, The next closest word is used. The next closest word is arrived by using the below steps.

- All words having hypernym information are extracted and vectorised using the embedding algorithm.
- The cosine distance is used to find the nearest word in the list of hypernyms vectors against the query vector.
- The terms are extracted for the nearest super-subordinate term, and the steps used for subject-related terms are followed.

5.2.4 Issues with Semantic Information Extraction

There were two instances for which the implementation failed to produce related terms. First, while querying for terms which are similar to words that are not in the extracted terminology list. Second, when querying words without hypernyms in the WordNet dataset. Since fasttext can handle Out of vocabulary vectors(OVV), we exploit its property to over the terminology querying to handle the query words which are not part of terminologies. In the first case, it is less complex as the query word vector is first extracted from the fasttext model, then the terms relating to the query word vector are found using cosine similarity. For the second case of contextual querying, it gets a little tricky and done in 3 steps as follows:

- First, the lemmas from WordNet are extracted and vectorized using the fasttext model.
- Second, the vector for the query word is also extracted using the same fasttext model.
- Next, vectors of lemmas, similar to the query word, are recognized using the cosine similarity between the measure.
- Finally, the vectors of lemmas, are summed up or averaged, and a list of similar terminologies are recognized using the same similarity measure.

Contextual Terms of "Narcotics"	Similarity Value	Contextual Terms of "Weapons"	Similarity Value
methadon hydrochloride	0.93	hook blade knife	0.93
methadrine	0.92	metal blade knife	0.93
heroin/methadone addiction	0.92	hook-blade knife	0.92
narcotic painkiller intake	0.91	wooden cylinder head	0.92
amphetamine addiction	0.91	hook knife	0.92
heroin/methadone	0.91	switch blade knife	0.92
methadone dosage	0.90	bone-handled knife	0.91
PCF marijuana	0.90	grooving knife	0.91
opiate methadone addiction	0.90	blade lever	0.91
methadon	0.90	four-inch blade switch blade	0.91
lidocaine	0.90	jammed hopper	0.91
morphine hydro-chloride	0.90	foot-long knife	0.91
daily heroin use	0.90	wooden hammer handle	0.91
morphine hydrochloride	0.90	rope handle	0.90
methadone addiction	0.90	gun metal	0.90
diacethyl morphine hydrochloride	0.90	bladed knife	0.90
added pure apomorphine	0.90	clamping sleeve	0.90
amphetamine overdose	0.90	wooden tamping stick	0.90
hydrochloride narcotic	0.90	boxcutter knife	0.90
amphetamine problem	0.89	pump-handle type lever	0.90
amphetamine pyribenzamine	0.89	low-hanging wire	0.90
quinine monohydrochloride	0.89	jack knife	0.90
marijuana metabolites	0.89	hatchet head	0.90
dihydromorphinone hydrochloride	0.89	spiked leather wrist band	0.90
fast-acting barbiturate	0.89	yellow-handled knife	0.90
ingested marijuana	0.89	fold-down knife	0.90
codeine hydrochloride	0.89	cutter head	0.90
ciacetyl morphine hydrochloride	0.89	gun type nozzle	0.90
amphetamine hydrochloride	0.89	push-button knife	0.90

Table 5.15: Top 30 Contextual Terminologies for "Narcotics" and "Weapons"

Contextual Terms of "Narcotics"	Similarity	Contextual Terms of "Weapons"	Similarity
controlled substance methamphetamine hydrochlo...	0.886	blade-type weapon	0.780
generic drug phenmetrazine hydrochloride	0.885	sawed-off weapon	0.765
depressant drug barbituric acid	0.872	bludgeon-type weapon	0.764
methadon hydrochloride	0.871	small-caliber weapon	0.757
substance methylenadioxy-methamphetamine	0.871	derringer weapon	0.756
phencyclidine hydrochloride	0.870	long-bladed knife	0.752
cocaine benzoylecgonine oxycodone diazepam	0.870	Dunn's weapon	0.750
diacetylmorphine hydrochloride	0.869	weapon knife	0.750
controlled substance methylenedioxy amphetamine	0.869	machete-type knife	0.747
morphine cocaine hydrochlorate	0.868	sharp weapon	0.744
amphetamine hydrochloride	0.863	caliber weapon	0.740
cocaine morphine hydrochlorate	0.863	Batley's weapon	0.740
meperidine hydrochloride	0.862	dangerous weapon knife	0.739
drug toxicity amphetamine	0.862	25-caliber weapon	0.738
fluphenazine chlorpromazine benztropine diphen...	0.860	Class II weapon	0.738
phenmetra-zine hydrochloride	0.860	weapon' kitchen knife	0.737
Diacetyl Morphine Hydrochloride	0.859	small caliber weapon	0.737
methamphetamine hydrochloride	0.859	large caliber weapon	0.734
ciacetyl morphine hydrochloride	0.859	five-inch-long hatchet knife	0.734
phenmetrazine hydrochloride	0.857	Boekhoff's pocketknife	0.734
diacetyl morphine hydrochloride	0.856	vicious weapon	0.732
controlled substance methylenedioxyamphetamine	0.854	four-bladed knife	0.732
injection haloperidol haloperidol decanoate lo...	0.854	gunman's weapon	0.731
morphine hydrochloride	0.853	Thompson's weapon	0.731
narcotic morphine sulphate	0.852	Henderson's weapon	0.730
cocaine pentazocine diazepam	0.850	I'M DONE A knife	0.730
psychotropic medication haloperidol decanoate	0.850	bayonet-style army knife	0.730
phencyclidine amphetamine	0.849	small weapon	0.730
morphine sulphate cocaine heroin	0.849	bayonet-style knife	0.730
nordiazepam oxazepam temazepam morphine oxycodone	0.847	Either weapon	0.729
one-half grain morphine sulfate	0.847	copper-jacketed projectile	0.729

Table 5.16: Top 30 Contextual Terminologies for "Narcotics" and "Weapons" using Pre-Trained Wiki-Embeddings

Chapter 6

Conclusion

In the preceding section, it has been shown that Automatic Term Recognition can be successfully performed and similar terms can be mined with respect to semantic and contextual relations. We have evaluated and demonstrated several methods for the automatic creation of a list of the most relevant key terms from Case Law Dataset. The results are favourable when analysed manually. The evaluation on the ground truth datasets is also comparable, although the state of the art performance is still relatively low in terms of precision and recall.

There are some key points to note about our study. It has been noted in the literature by Castellvi et al. that "POS disambiguation is one of the most important error sources" in ATR. As described in the Discussion and evaluation section, the default NLTK POS Tagger did not provide the required level of accuracy in tagging. For instance, 'drinking' was recognized as a noun, although its correct parts of speech is a 'verb'. It is difficult to accurately tag words even by human as words have tendencies of ambiguity like 'cook' which can be a noun and verb. Long term dependencies of a word can aid identify POS tags.

Term Recognition is highly domain-specific. A method used on one corpus may not necessarily give the same level of precision and recall. This is shown in the analysis section and also consequently showing personalized filters have better F1 score than open and closed filters.

Finally, a novel method was presented to query similarity based on semantic and contextual relations using an external source. Such a source can be augmented with the terms generated using the hypothesis defined in this literature.

6.1 Future Implementation

Future work will include the addition of external features from the datasets domain resource. Such resources can provide more domain-specific background information.

Alternate POS tagger will be used to find POS sequences, and the results can be compared in order to determine if this new tagger will increase the accuracy of POS tagging. Also, multiple taggers can be used to assign a tag detected by the voting.

Text clustering algorithms like Affinity Propagation[Frey and Dueck, 2007] seems a promising approach to be explored. Clustering can be applied over the trained word

vectors of the corpus in question. This would group related words in separate categories. The number of clusters is automatically learnt in Affinity propagation. The words in each cluster would represent relatedness amongst the group leading to one-shot work.

Bibliography

[Almeida and Xexéo, 2019] Almeida, F. and Xexéo, G. (2019). Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*.

[Astrakhantsev, 2018] Astrakhantsev, N. (2018). Atr4s: toolkit with state-of-the-art automatic terms recognition methods in scala. *Language Resources and Evaluation*, 52(3):853–872.

[Astrakhantsev et al., 2015] Astrakhantsev, N. A., Fedorenko, D. G., and Turdakov, D. Y. (2015). Methods for automatic term recognition in domain-specific text collections: A survey. *Programming and Computer Software*, 41(6):336–349.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Bojanowski et al., 2017] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[Braslavskii and Sokolov, 2008] Braslavskii, P. and Sokolov, E. (2008). Comparison of five methods for recognition of terms of arbitrary length. *Komp'yuternaya lingvistika i intellektual'nye tekhnologii (Computational Linguistics and Intellectual Technologies)*, (7):14.

[Camacho-Collados and Pilehvar, 2018] Camacho-Collados, J. and Pilehvar, M. T. (2018). From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

[Church and Gale, 1999] Church, K. and Gale, W. (1999). Inverse document frequency (idf): A measure of deviations from poisson. In *Natural language processing using very large corpora*, pages 283–295. Springer.

[Dagan and Church, 1994] Dagan, I. and Church, K. (1994). Termight: Identifying and translating technical terminology. In *Fourth Conference on Applied Natural Language Processing*, pages 34–40.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Faralli and Navigli, 2013] Faralli, S. and Navigli, R. (2013). Growing multi-domain glossaries from a few seeds using probabilistic topic models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 170–181.

[Fedorenko et al., 2014] Fedorenko, D., Astrakhantsev, N., and Turdakov, D. (2014). Automatic recognition of domain-specific terms: an experimental evaluation. *Proceedings of the Institute for System Programming*, 26(4):55–72.

[Firth, 1957] Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

[Frantzi et al., 2000] Frantzi, K., Ananiadou, S., and Mima, H. (2000). Automatic recognition of multi-word terms:. the c-value/nc-value method. *International journal on digital libraries*, 3(2):115–130.

[Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976.

[Goldhahn et al., 2012] Goldhahn, D., Eckart, T., and Quasthoff, U. (2012). Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*, volume 29, pages 31–43.

[Gomaa et al., 2013] Gomaa, W. H., Fahmy, A. A., et al. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18.

[Harris and Harris, 2010] Harris, D. and Harris, S. (2010). *Digital design and computer architecture*. Morgan Kaufmann.

[Hasan and Ng, 2014] Hasan, K. S. and Ng, V. (2014). Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273.

[Heylen and De Hertog, 2015] Heylen, K. and De Hertog, D. (2015). Automatic term extraction. *Handbook of Terminology*, 1(01).

[Jackson et al., 1998] Jackson, P., Al-Kofahi, K., Kreilick, C., and Grom, B. (1998). Information extraction from case law and retrieval of prior cases by partial parsing and query generation. In *Proceedings of the Seventh international Conference on information and Knowledge Management*, pages 60–67.

[Jones, 1972] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

[Joulin et al., 2016a] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016a). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

[Joulin et al., 2016b] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016b). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[Judea et al., 2014] Judea, A., Schütze, H., and Brügmann, S. (2014). Unsupervised training set generation for automatic acquisition of technical terminology in patents. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical Papers*, pages 290–300.

[Justeson and Katz, 1995] Justeson, J. S. and Katz, S. M. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural language engineering*, 1(1):9–27.

[Kageura and Umino, 1996] Kageura, K. and Umino, B. (1996). Methods of automatic term recognition: A review. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 3(2):259–289.

[Kim et al., 2003] Kim, J.-D., Ohta, T., Tateisi, Y., and Tsujii, J. (2003). Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

[Koehn, 2005] Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.

[Krapivin et al., 2009] Krapivin, M., Autaeu, A., and Marchese, M. (2009). Large dataset for keyphrases extraction. Technical report, University of Trento.

[Kusner et al., 2015] Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. In *International conference on machine learning*, pages 957–966.

[Lang, 1995] Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

[Liu et al., 2015] Liu, Y., Liu, Z., Chua, T.-S., and Sun, M. (2015). Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Loper and Bird, 2002] Loper, E. and Bird, S. (2002). Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

[Lossio-Ventura et al., 2013] Lossio-Ventura, J. A., Jonquet, C., Roche, M., and Teisseire, M. (2013). Combining c-value and keyword extraction methods for biomedical terms extraction. In *LBM: Languages in Biology and Medicine*.

[Loukachevitch and Nokel, 2013] Loukachevitch, N. and Nokel, M. (2013). An experimental study of term extraction for real information-retrieval thesauri. In *Proceedings of Terminology and Artificial Intelligence Conference TIA-2013*, pages 69–78. Citeseer.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.

[Maxwell and Schafer, 2008] Maxwell, K. T. and Schafer, B. (2008). Concept and context in legal information retrieval. In *JURIX*, pages 63–72.

[Medelyan and Witten, 2008] Medelyan, O. and Witten, I. H. (2008). Domain-independent automatic keyphrase indexing with small training sets. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Milios et al., 2003] Milios, E., Zhang, Y., He, B., and Dong, L. (2003). Automatic term extraction and document similarity in special text corpora. In *Proceedings of the sixth conference of the pacific association for computational linguistics*, pages 275–284. Citeseer.

[Miller, 1995] Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Nakagawa, 2001] Nakagawa, H. (2001). Experimental evaluation of ranking and selection methods in term extraction. *Bourigault D, L'Homme M.-C., Jacquemin C.(éd.), Recent advances in computational terminology, John Benjamins Publishing Company*, pages 303–26.

[Neelakantan et al., 2015] Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2015). Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[Pazienza et al., 2005] Pazienza, M. T., Pennacchiotti, M., and Zanzotto, F. M. (2005). Terminology extraction: an analysis of linguistic and statistical approaches. In *Knowledge mining*, pages 255–279. Springer.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Prasath and Öztürk, 2011] Prasath, R. and Öztürk, P. (2011). Finding potential seeds through rank aggregation of web searches. In *International Conference on Pattern Recognition and Machine Intelligence*, pages 227–234. Springer.

[QasemiZadeh and Handschuh, 2014] QasemiZadeh, B. and Handschuh, S. (2014). The acl rd-tec: A dataset for benchmarking terminology extraction and classification in computational linguistics. In *Proceedings of the 4th International Workshop on Computational Terminology (Computerm)*, pages 52–63.

[QasemiZadeh and Schumann, 2016] QasemiZadeh, B. and Schumann, A.-K. (2016). The acl rd-tec 2.0: A language resource for evaluating term extraction and entity recognition methods. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1862–1868.

[Scott and Tribble, 2006] Scott, M. and Tribble, C. (2006). *Textual patterns: Key words and corpus analysis in language education*, volume 22. John Benjamins Publishing.

[Wang et al., 2019] Wang, B., Wang, A., Chen, F., Wang, Y., and Kuo, C.-C. J. (2019). Evaluating word embedding models: methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8.

[Weeds, 2003] Weeds, J. E. (2003). *Measures and applications of lexical distributional similarity*. PhD thesis, University of Sussex.

[Wermter and Hahn, 2006] Wermter, J. and Hahn, U. (2006). You can't beat frequency (unless you use linguistic knowledge): a qualitative evaluation of association measures for collocation and term extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 785–792. Association for Computational Linguistics.

[Wong, 2009] Wong, W. (2009). Determination of unithood and termhood for term recognition. In *Handbook of research on text and web mining technologies*, pages 500–529. IGI Global.

[YUAN et al., 2015] YUAN, J.-s., ZHANG, X.-m., and LI, Z.-j. (2015). Survey of automatic terminology extraction methodologies. *Computer Science*, (8):3.

[Zhang and Wu, 2012] Zhang, C. and Wu, D. (2012). Bilingual terminology extraction using multi-level termhood. *The Electronic Library*.

[Zhang et al., 2016] Zhang, Z., Gao, J., and Ciravegna, F. (2016). Jate 2.0: Java automatic term extraction with apache solr. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2262–2269.

[Zhang et al., 2018] Zhang, Z., Gao, J., and Ciravegna, F. (2018). Semre-rank: Improving automatic term extraction by incorporating semantic relatedness with personalised pagerank. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):1–41.

[Zhang et al., 2008] Zhang, Z., Iria, J., Brewster, C., and Ciravegna, F. (2008). A comparative evaluation of term recognition algorithms. In *LREC*, volume 5.

Appendix A

Generators

In Python, implementing an iterator object is an expensive task. A class with *_iter_()* and *_next_()* methods, keep track of internal states and raise 'stop iteration' when no values are returned. This is lengthy and counter-intuitive. It can be handled more efficiently with the use of generators. Python generators are a simple way of creating iterators. All the work we mentioned above is automatically created when a generator is defined.

Generators are as easy as defining a normal function. It may contain either *yield* *return*. The only difference between them is that the *return* statement terminates the function entirely whereas, *yield* statement pauses the function saving all its states and later continues from there on successive call. A common use case of the generator is to work with data streams or large files, like CSV files. These text-files separate data into columns by using commas. When large files are accessed without generators, a large file is loaded at once causing it to occupy the memory and slowing down the system if there is no enough buffer space or crashing the system if the size is less than available memory, whereas a generator would yield each without causing memory errors.

Model dimension Reduction Techniques

The pre-trained Wikipedia binary models mentioned in literature consumes 9 GB of memory when loaded. This fact makes it hard to use subsequent implementation of logic over the models. There are two main reasons why binary models occupy so much memory. First, the binary model not only carries weights for words but also for n-grams but also weights for translating vectors back to words. These vectors are also negative vectors which are used for additional training. Second, the model vocab and n-gram matrices are huge; they contain 2M vectors each.

To mitigate the first problem, dropping the training weights would suffice. Also, from the vocabulary matrix, the less frequent words can be dropped. With the n-gram matrix, the model does a hashing trick. The n-gram hashes are mapped by the remainder of the division by matrix size. So the hashes need to be calculated again to remap the indexes.

Another way is to reduce the dimension of the vectors, which is implemented in this literature. Details can be found in the fasttext documentation [1]

[1] url https://fasttext.cc/docs/en/crawl-vectors.html

Appendix B

Sample ATR Python Implementation

```
import nltk
from nltk.tokenize import WhitespaceTokenizer
import lzma, json
import sys, os, time
wt = WhitespaceTokenizer()

#Functions
def tokenize_data(corpus_file):
    with lzma.open(corpus_file) as in_file:
        for line in in_file:
            case = json.loads(str(line, 'utf8'))
            tokenized_para = []
            for opinion in case['casebody']['data']['opinions']:
                sentences = sent_tokenize(opinion['text'])
                for sent in sentences:
                    tokenized_para.append(list(filter(None,remove_punctua
            tokenized_name=list(filter(None,(wd(case['name']))))
            yield tokenized_para, tokenized_name

def tag_pos(tokenize_list, grouped=False):
    if grouped:
        return nltk.pos_tag_sents(tokenize_list)
    else: return nltk.pos_tag(tokenize_list)

def extract_np(parsed_sent):
  for subtree in parsed_sent.subtrees():
    if subtree.label() == 'NP':
        yield '_'.join(word for word, tag in subtree.leaves())
        # yield (' '.join(word for word, tag in subtree.leaves()), ' '.jo

def chunker(tagged_sent, grouped=False):
    grammar = r"""
____NP:
```

```
    # ATR4s - Nikitha Astrakhantsev
        #{(<NN(S)?>|<JJ>|<NNP>|<NN(S?)><IN>)*(<NN(S)?>)}
    #Frantzi & Justeson #Open
        # {<NN.*>+<NN.*>}
        # {(<JJ>|<NN.*>)+(<NN.*>)}
        # {((<JJ>|<NN.*>)+|((<JJ>|<NN.*>)*(<NN.*><IN>)?)(<JJ>|<NN.*>)*)<NN.*>
    #termight (Closed)
        # {<NN.*>+}
    #NLTK
        # {<DT>?<JJ>*<NN.*>}
    #ATR - Personalized
        {<NNP>{1,3}<NN?>?}
        {(<JJ>|<NN>)<NN?>{1,3}}
        {<DT>?<JJ>*<NN>}
        {<DT>?<NN>+}
    """
    cp = nltk.RegexpParser(grammar)
    if grouped:
        parsed_sent = list(map(cp.parse, filter(None, tagged_sent)))
        return list(filter(None, map(extract_np, parsed_sent)))
    else:
        parsed_sent = cp.parse(tagged_sent)
        return extract_np(parsed_sent)

def linguistic_filter(corpus_file):
    count= 0
    #Tokanize Data generator
    print('process started')
    tokenize_data_gen = tokenize_data(corpus_file)
    for tokenized_para, tokenized_name in tokenize_data_gen:
        #POS- Tagging
        tagged_para = tag_pos(tokenized_para,grouped=True)
        tagged_name = tag_pos(tokenized_name)
        #Chunking
        chunked_para=[term for term in map(list, chunker(tagged_para, grouped
        chunked_name=[term for term in chunker(tagged_name)]
        yield chunked_para, chunked_name

#Class for related terms search
import numpy as np
import functools
from functools import reduce
import operator
from nltk.corpus import wordnet as wn
import nltk
```

```
nltk.download('wordnet')

class Term2Vec:

    def __init__(self, name):
        self.name = name

    def fit(self, model, terms, tokenwise = False):
        def unNest(terms):
            ''' recursively flattern nested lists '''
            if not (any(isinstance(i, list) for i in terms)):
                return terms
            else:
                terms = functools.reduce(operator.iconcat, terms, [])
                return unNest(terms)
        self.model = model
        self.terms = list(set(unNest(terms)))
        if not tokenwise:
            self.term_vec= np.array([model.get_word_vector(x) for x in se
            self.idx_words= dict(enumerate(self.terms))
            self.words_idx = dict(zip(self.idx_words.values(),self.idx_wo
        else:
            self.term_vec = np.array([np.array([model.get_word_vector(w)
        return self

    def get_vec(self, term):
        return self.term_vec[term]

    def cosine_sim(self, A,B):
        return np.divide(B.dot(A),np.multiply(np.linalg.norm(A),np.linalg

    def get_similar_terms(self, term, N=10):
        if term not in self.words_idx.keys():
            ov_vec = self.model.get_word_vector(term)
            sim_mat = self.cosine_sim(ov_vec,self.term_vec)
        else:
            id= self.words_idx[term]
            sim_mat = self.cosine_sim(self.term_vec[id],self.term_vec)
        return [(self.idx_words[n], sim_mat[n]) for n in list(np.argsort(

    def get_contextual_terms(self, term, N=10):
        #Get Embedding for wordnet words
        wordnet_vec= np.array([self.model.get_word_vector(x) for x in wn.
        idx_wordnet= dict(enumerate(wn.all_lemma_names()))
        wordnet_idx = dict(zip(idx_wordnet.values(),idx_wordnet.keys()))
```

```
        wordnet_vec.shape

        def getLemmas(word):
            synonyms = []
            for syn in wn.synsets(word):
                for h in syn.hyponyms():
                    for l in h.lemma_names():
                        synonyms.append(l)
            return list(set(synonyms))

        def get_hypo_vectors(self, word):
            hypo_list = getLemmas(word)
            if len(hypo_list) ==0:
                ov_vector = self.model.get_word_vector(word)
                sim_mat = self.cosine_sim(ov_vector,wordnet_vec)
                hypo_vec = np.array(wordnet_vec[np.argsort(-1*sim_mat)[:10]].
            else:
                hypo_vec= np.array([self.model.get_word_vector(x) for x in hy
            return hypo_vec

        hypo_vec = get_hypo_vectors(self, term)
        sim_mat = self.cosine_sim(hypo_vec.mean(axis=0),self.term_vec)
        return [(self.idx_words[n], sim_mat[n]) for n in list(np.argsort(-1*s

    def get_contextual_subjects(self,term,N=10):
        def getLemmas(word):
            synonyms = []
            for syn in wn.synsets(word):
                for h in syn.hypernyms():
                    for l in h.lemma_names():
                        synonyms.append(l)
            return list(set(synonyms))

        def get_hyper_vectors(self, word):
            hyper_list = getLemmas(word)
            hyper_vec= np.array([self.model.get_word_vector(x) for x in hype
            return hyper_vec

        hyper_vec = get_hyper_vectors(self, term)
        sim_mat = self.cosine_sim(hyper_vec.mean(axis=0),self.term_vec)
        return [(self.idx_words[n], sim_mat[n]) for n in list(np.argsort(-1*s

if __name__ =='__main__':
        length_para=0
        obj_dir = '/content/gdrive/path/'
```

```
    list_text_file = os.path.join(obj_dir, 'opinion.txt')
tm=start_time= time.time()
linguistic_candidates_gen =  linguistic_filter(corpus_file)
count=0
terms_para=[]
terms_name= []
for para, name in linguistic_candidates_gen:
    terms_para.append(rem_specialchars(para))
    terms_name.append(name)
    length_para+=getSizeOfNestedList(para)
    count+=1
    if count%50 == 0: tm=(time.time()-start_time)/count
    eta = time.strftime("%H:%M:%S",time.gmtime(tm*(183146-count)))
    sys.stdout.write('\r'+'Progress:     '+'{:.1f}%'.format(count*100/
    '     Doccuments Processed: '+str(count)+\
    '     Token Length: '+str(length_para)+\
    '     ETA: '+str(eta))
```

List of Abbreviations

ACL	Association for Computer Linguistics
ATE	Automatic Term Extraction
ATR	Automatic Term Recognition
AvP	Average Value Precision
CAP	Case Access Project
CBOW	Collective Bag of Words
CD	Case Decisions
Collocation	A pair or group of words that are habitually juxtaposed. E.g., Win-Win, Heavy-Drinker.
Contractions	The process of shortening a word by combination or elision. E.g., "couldn't've" - "could not have"
F1-Score	A weighted harmonic mean between Precision and Recall
FN	False Negatives
FP	False Positives
NLTK	Natural Language TookKit
NP	Noun Phrases
OVV	Out Of Vocabulary Vectors
POS	Parts of Speech
POST	Parts of Speech Tagging
R-IDF	Residual - Inverse Document Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
TM	Terminology Mining
TP	True Positives
WMD	Word Movers Distance

List of Algorithms

1 Term Recognition Model 28
2 Semantic Relation Model 32
3 Context Relation Model 33
4 Implementation of the testing 45

List of Figures

3.1 Noun Phrases based on POS Tags 25
3.2 Term Generator 27
3.3 Semantically Related Terms 31
3.4 Contextual Related Terms 32

5.1 Graph of POS Tag Frequencies 50
5.2 Graph of Candidate Terms Tag Pattern Frequencies 55

List of Tables

2.1 Survey of Literature in Automatic Term Recognition 10
2.2 Table of comparison for different embedding models 17
2.3 Text Similarity Measures 20

3.1 Parts of Speech tags in NLTK 23
3.2 Word embeddings Model Characteristics 30

4.1 Colab System Specifications 35
4.2 Dataset Statistics 36
4.3 Linguistic Filters applied on Ground Truth Datasets 38
4.4 Performance of Statistical Filter over Genia Dataset 39
4.5 Performance of Statistical Measures over ACL2 Dataset 40
4.6 Performance of Statistical Filter over Patents Dataset 41
4.7 List of the 20 newsgroups, partitioned according to the subject matter . . . 44
4.8 Top 5 Word Similarities and Analogies using multiple Embedding models . 45

5.1 CAP Dataset Statistics 48
5.2 Count of Tags in the First 10K Case Decision 50
5.3 Sample Terms of ATR Tool Linguistic Filter [Astrakhantsev, 2018] 51
5.4 Sample Terms of "Closed" Linguistic Filter 52
5.5 Sample Terms of "Open" Linguistic Filter 53
5.6 Sample Terms NLTK Linguistic Filter 53
5.7 Sample Terms of "Personalised" Linguistic Filter 54
5.8 Top and Bottom 30 terms in TF-IDF 57
5.9 Top and Bottom 30 terms in Residual - IDF 58
5.10 Top and Bottom 30 terms in C-Value 59
5.11 Top and Bottom 30 terms in C/NC - Value 60
5.12 Top and Bottom 30 terms in ComboBasic 61
5.13 Top 30 similar Values for "Narcotics" and "Weapons". 63
5.14 Top 30 similar Values for "Narcotics" and "Weapons" using Pre-Trained Wiki-Embeddings 64
5.15 Top 30 Contextual Terminologies for "Narcotics" and "Weapons" 66
5.16 Top 30 Contextual Terminologies for "Narcotics" and "Weapons" using Pre-Trained Wiki-Embeddings 67

Acknowledgements

With boundless love and appreciation, the graduate candidate would like to extend his heartfelt gratitude and appreciation to the people who helped him bring this study into reality. The candidate would like to extend his profound gratitude to the following:

Foremost his adviser, Prof. Alfio Ferrara whose expertise, consistent guidance, ample time spent and consistent advice that helped him bring this study into success.

To the Panel of Examiners, Prof. Nicolo' Antonio Cesa Bianchi, Prof. Federico Avanzini, Prof. Laura Anna Ripamonti, Prof. Andrea Trentini, Prof. Stefano Montanelli.

To all his course professors Prof. Alberto Ceselli, Prof. Roberto Sassi, Prof. Stefano Ferrari, Prof. Simone Agazzi, Prof. Sara Foresti, Prof.Gabriella Trucco, Prof.Valentina Ciriani, Prof. Fabio Scotti, Prof. Vincenzo Piuri, and Prof. Ernesto Damiani.

To the tech gurus Abhinav Anand, Rivolta Massimo for their constructive comments, suggestions and critiques.

To his lab mates: Francesco Polvere, Marco Santinelli, Mattia Falduti for the stimulating discussions and for all the fun they have had in the last few months.

Also thank all his friends at University of Milan: Naveen H S, Cassiano Vailati, Sina Mohammadzadeh, Blerd Foniqie, Thomas Augier, Matteo Gandossi, Luca Diedolo, Raffaele Vinci, Chiara, Giuseppe Vadrucci.

Last but not least, He would like to thank his family: His parents Natividа and Dr Manuel Fernandes for all the love and support throughout his life.