Université de Liège - Faculté des Sciences Appliquées

Travail de fin d'études réalisé en vue de l'obtention du grade de master "en science des données"

by Judicael POUMAY

Term extraction from domain specific texts

Supervised by Ashwin ITTOO

Academic year 2018-2019

Contents

1 Introduction **2**
- 1.1 Background . . . 2
- 1.2 Literature . . . 2
- 1.3 Challenges . . . 2
- 1.4 Our contribution . . . 2
- 1.5 Thesis structure . . . 3

2 Literature review **4**
- 2.1 Linguistic approach . . . 4
- 2.2 Statistical approach . . . 6
  - 2.2.1 Unithood . . . 7
  - 2.2.2 Termhood . . . 8
  - 2.2.3 TF-IDF . . . 10
- 2.3 Hybrid Approach . . . 10
- 2.4 Limitations and challenges of past studies . . . 11

3 Methodology **13**
- 3.1 Pre-processing . . . 13
- 3.2 Term extraction : principles . . . 14
- 3.3 Term extraction : implementation . . . 16
  - 3.3.1 The full algorithm . . . 16
  - 3.3.2 N-grams extraction . . . 17
  - 3.3.3 Supergrams extraction . . . 18
  - 3.3.4 Selecting results . . . 20
- 3.4 Implementation decision for the different measures . . . 20
- 3.5 Abbreviation extraction . . . 20
- 3.6 Memoisation . . . 21

4 Experiments and results **22**
- 4.1 Data . . . 22
- 4.2 Demonstrating POS tagging inefficiencies . . . 23
- 4.3 Comparing measures . . . 25
  - 4.3.1 Termhood : Frequency and C-value . . . 25
  - 4.3.2 Unithood : Jaccard, PMI, Dice, and T-score . . . 26
  - 4.3.3 Noise : ISW . . . 27
  - 4.3.4 Extracting terms using our CPS^2 measure . . . 27
- 4.4 Supergrams . . . 28
- 4.5 Abbreviation extraction . . . 29
- 4.6 Performance analysis . . . 29

5 Conclusion **31**

1 Introduction

1.1 Background

Like many other fields of data science, natural language processing is becoming a more important part of our lives. This technological revolution of machine learning is unlike any other. While big steam machines made such noise and took such huge amount of space that they were not ignorable. Intelligent systems come to most people in the form of subtle convenience in their daily life. But for big companies they can prove to be incredible assets if implemented correctly.

Natural language processing contains many interesting sub-fields such as information retrieval or automatic translation. In this thesis, we will focus on terminology extraction. The idea of terminology extraction is to derive from a corpus a set of specific concepts. A term is, therefore, a sequence of words which references a specific concept in a particular domain. For example, an "asset" is a resource but a "financial asset" is an economic resource and so on.

The reason for extracting term can be many. We can use term extraction as a way to quickly understand what a text is about by extracting key phrases. This can make search engine query faster or help summarize a document. Term extraction is also used to create thesauri or populate ontologies.

1.2 Literature

Several algorithms exist in the literature for terminology extraction, but they suffer from various limitations. First, many of these algorithms have issues extracting long terms ([1],[2]), especially linguistic methods which suffer from a complexity problem. Second, many of these algorithm need some kind of supervised learning wether it be minimal or total [3]. Finally, many focus on particular domains such as bio-medical while other domain such as financial regulation are overlooked. Thus as a domain corpus, we rely on financial regulation documents published on Eur-Lex (click here for a sample).

1.3 Challenges

Thus, the main challenges we face in term extraction are the following. First is sparsity, if a term appear only a few times in a big document, it is more difficult to extract. On the other hand some constructs such as "for example" may appear frequently while being irrelevant. Thus, we must be careful when using frequency as a measure of relevancy. The third problem is long terms, and current methods are either incapable or have huge difficulties with the extraction of terms made of more than three words (e.g. [1],[2]). Financial regulatory texts are especially interesting in their nature as they contain many long terms. Finally, contrary to domain such as the bio-medical [4] which appreciates a huge ontological repository of knowledge which is often helpful for term extraction. European financial legislative text do not enjoy such resources. This makes the task of producing and evaluating an algorithm more difficult.

1.4 Our contribution

The problem that is at the core of this thesis is, therefore: how do we extract relevant information from huge volume of highly technical and domain specific texts? To answer this question, we propose a novel method that alleviates the limitations of existing algorithm. It is based on an n-gram extraction approach and allied with a newly proposed measure so-called CPS^2 that we have crafted. Moreover, we used the extracted n-grams (n =2,3,4) to derive

'supergrams' which allows us to precisely detect long terms (up to 10 words). Indeed, we can extract long terms by concatenation of smaller terms that we have detected. In the end, our algorithm outputs a set of n-grams (n =2,3,4) and a set of supergrams extracted from a given text.

Our algorithm is unsupervised and robust to noise thanks to a careful filtering phase. It is also fast with the help of memoïsation to cache already computed and often used results. Implementing only such a statistical approach, experimental evaluation shows that our method outperforms a state-of-the-art linguistic baseline on the financial regulatory corpus. Finally, we have also crafted a second auxiliary algorithm for abbreviation extraction.

1.5 Thesis structure

In the following section we will perform a review of the current literature. Next in section 3 we will describe our novel algorithm. We will continue with a section presenting our results and then end with the conclusion.

2 Literature review

Terminology extraction (TE) represents a subset of information extraction (IE) which itself is a subset of natural language processing (NLP). It is the process of detecting, extracting, and ranking terms from documents. A term is a sequence of words that acts as a semantic unit which refers to a specific concept in a specific domain, for example: "international monetary fund". The task of TE has found widespread application in ontology learning ([5],[6]) and artificial intelligence systems. In our case, the TE algorithm we developed belongs to a more complex pipeline dedicated to automatic thesaurus learning.

Two fundamental approaches exist for extracting terms: statistical and linguistical (See [7]). While the latter is more concerned about numerical measurements to rank and filter candidate terms. The former uses syntactic knowledge to recognize and extract candidate terms using pattern seeking algorithms. In practice, a mix of both approaches is frequently implemented and may lead to better results at the expense of speed and complexity.

But linguistical approaches are typically limited in the size of the term extracted (often 3-gram maximum) and hybrid method thus inherits from this limitation. And while such technique demonstrate impressive results for small terminology extraction. This means these methods are unsuited for deriving longer and more complex terminology. And since our domain of interest, financial regulatory documents, contains such terminology we had to forgo linguistical methods.

Moreover, the notion of term is presently unclear. Indeed, human language is messy and made up of often arbitrary rules. Thus, it is challenging to devise algorithms capable of grasping the concept of termhood since human themselves have difficulty accurately defining what a term is. Therefore, the definition of what is a term is often decided on a project-per-project basis with the help of domain experts. In our case for example, financial regulatory text contains long and complex terms that are made up of up to 10 words or more. Domain expert in this field provided us with an example of such term:

> "financial assets designated at fair value through profit or loss".

2.1 Linguistic approach

Linguistical approaches employ the syntactic structure of a text to extract candidate terms. In particular, part-of-speech (POS) tagging methods such as [8] allows us to associate each word in a sequence with a syntactic tag such as noun (N), adjective (A), adposition(AP), verb (V), Preposition (P) etc.

In [9], it is claimed that such syntactic analysis is sufficient for terminology extraction. This study was conducted for the Research Development Division of Electricité de France (RDDEF) with the goal of keeping their thesaurus up to date in a constantly evolving field. Thus, the corpus used for developing their algorithm is made french documents provided by the RDDEF. Note that the study does not contain an experiment nor a results section.

The algorithm they propose (LEXTER) works in two stages. Firstly, POS-tagging is used to mark frontiers elements to extract maximal length noun phrases. More precisely, the idea is to use negative knowledge about terms, that is they assume that terms do not contains verbs or some prepositions. These elements are then detected and replaced by frontier markers. Maximal length noun phrases delimited by these markers can thus be extracted. Secondly, on these extracted phrases are applied syntactic rules to extract sub-component using POS-tagging. These rules assume that terminological units are made of a specific kind of grammatical elements,

for example: [N P N]. By applying this kind of rules to maximal length noun phrases, they were able to extract terms. The terms extracted by LEXTER are limited in their content, no verbs or prepositions (with two exceptions). Consequently, the final terminology extracted does not contain long or complex terms.

In [1], they show an hybrid and unsupervised method for information retrieval based on the extraction of sub-components from complex noun phrases. They argue that a linguistic approach is more effective as a way to assist statistical analysis of the data by filtering out impossible term structure such as [A A]. A linguistic approach can also rank possible structure, they argue that term made of [N N] should be given more weight than [A N] terms. In their case, they also used syntactic knowledge to generate possible term from a noun phrase. More precisely, from the phrase "the quality of surface of treated stainless steel strip" they were able to generate "treated strip", "stainless steel", "stainless strip", and "steel strip" or "strip surface", "surface quality", and "strip quality". Indeed, being able to generate new concepts from existing one has a lot of potential for information retrieval as it allows for more relevant queries to extract the same documents.

To help with the process of sub-component generation and noun phrase parsing, they use POS-tagging to discover terms using the following patterns: [N N], [A N], [Term N], [A Term], and [N Term]. Interestingly, three of these patterns are recursive. Indeed, a term t can be extracted using the pattern [N N], this term t can then be re-used to extract longer terms containing it using these recursive patterns: [Term N], [A Term], and [N Term].

The lexicon of terms derived from these patterns is used to find association between the elements of noun phrases. This is achieved by iteratively applying the patterns on a noun phrase until no more pattern can be found. For example, the phrase "general purpose high performance computer" leads to this set of association "[[general=purpose]=[[high=performance]=computer]]" by linking the the more reliable terms first, here "high performance". This entails that "general purpose" and "high performance computer" are two terms that are linked but "purpose high" is not a term. Once again this work did not focus on the extraction of long term but rather the discovery of smaller lexical units. Even though their theoretical analysis is relevant to this review, their experimentation focuses on the task of information retrieval and thus the results are not relevant and won't be discussed.

In [10], POS is used to extract candidate terms which are then filtered using a statistical analysis. In this study they also use traditional syntactic patterns such as [A N] or [N N] for English text, although they also look for the pattern [N "of" N]. But the study also focuses on French texts which contains different patterns: [N A], [N N], [N "de" N], [N P N].

Moreover, they explain that three basic operations can be applied on terms. Overcomposition, which simply corresponds to the concatenation of smaller terms to create bigger ones (e.g. "international monetary funds" is made of "monetary funds" and "international funds"). Modification, which is the process of adding a modifier to a term, mainly adjectives. Coordination, which takes two related concept to form a new one: "packet disassembly" and "packet assembly" become "packet assembly/disassembly".

Their study thus focuses on the extraction of small terms (two words not counting stopwords) using POS-tagging and the derivation of more complex words through these operations. To reduce noise, they use statistical analysis of the candidate terms extracted and filter out bad elements using their frequency. They used an existing terminology database EURODICAUTOM that they augmented by hand to test their results on French and English corpora. They reached 80% precision on the top-500 extracted terms and 70% for the top-1000. Again this method does not allow for the extraction of long/complex terminology, although their analysis of the

possible operations on terms is insightful. Indeed, our system for extracting long term is based on overcomposition of smaller terms.

POS-based approach can also be useful as a pre-processing step to filter out stop-words and unlikely candidate (as in [9] or [1]). Stop-words are words which do not convey a lot of valuable semantic information. Examples of stop-words are "the", "that", "I", etc. Such elements can be detected using POS tagging with tags such as articles, prepositions, and pronouns. In our case we used a standard pre-processed list of stop-words of the English language provided by NLTK [11].

Limitations We argue that although POS-based approaches lead to good results for simple terminology extraction. Such methods are not capable of extracting long and complex terminology without leading to overly complex and unmanageable systems. Indeed, as we include more complex rules, the model becomes rigid and difficult to maintain. While simpler rules prevent us from extracting more complex terms.

For example, if one wishes to design patterns to extract financial regulatory terms from our corpus, one of the pattern will be: [A N V AP N AP N CC N][1]. Which would extract the term: "financial assets designated at fair value through profit or loss" and probably only 2-3 others like it. But as can be expected, the longer and more complex patterns become and the more restrictive and difficult to craft they get. That is, the more complex the pattern the more time will be needed to craft it and the less numerous the term extracted will be. Moreover, this is only one of the many pattern that we would need to extract the complex terminology hidden in our corpus. Indeed, we would need several patterns like this and we would have to generalize them. Therefore, we argue that trying to craft rules manually for complex multi-word term extraction is an unviable method because it is laborious, restrictive, and not maintainable.

Consequently, we conclude that POS-tagging based term extraction remain effective tools to extract small and simple terminology. But if we wish to extract longer and more complex terms, we require a more flexible solution.

Moreover POS tagger uses stochastic methods or rule-based approach [8] to classify tokens. Although taggers typically have a high accuracy of 95-99%, this should prompt the reader that POS-tagger are also learnt system. And since terminology extraction is also prone to errors, we suddenly realise that a linguistic approach is a two-stage system. This means that error rates compound on each other in a multiplicative manner which represent another issue with linguistical approaches. This issue actually affects any multi-stage methods, and this is why we argue that simpler approaches are more desirable.

Besides, a linguistical approach alone is usually not sufficient to measure the relevancy of the term extracted. This is why it is often allied with a statistical approach which provides such measures of relevancy.

2.2 Statistical approach

Statistical approaches are ordinarily implemented to rank term extracted using linguistical approach. A fully statistical approach is also possible and is what we will be presenting in this thesis.

[1] CC is for coordinating conjunction.

After candidate terms are extracted using a linguistical or statistical approach. A statistical analysis of the candidates will provide a set of measures of relevancy. These measures fall into two categories: termhood and unithood measures [12].

2.2.1 Unithood

Unithood measures capture the likelihood that a sequence of words occurs together more often than by themselves. This is, therefore, a measure of how much a sequence of words act as a single unit throughout the document. Indeed, if a sequence of words mostly appears together, we may treat it as one single unit. For example the term "ad hoc" is made of two words but they are almost never used by themselves, therefore it acts as a single unit. It is self-evident that being able to measure unithood is an important step for term extraction. And measuring unithood can be done using the following measures:

Point-wise mutual information First defined in information theory [13], mutual information is intuitively a good contender to measure unithood. It is a measure of association between multiple elements, in this case the elements are words. It measures the information contained in the association of these words compared to the words themselves. It can be positive or negative but outputs zero for sequence of independent words, that is words that do not appear together.

It is most often defined for the bivariate case:

$$PMI(w_1, w_2) = log(\frac{P(w_1, w_2)}{P(w_1) * P(w_2)}) = log(\frac{f(w_1, w_2) * N}{f(w_1) * f(w_2)})$$

But it can easily be generalized to the multivariate case:

$$PMI(w_1, ..., w_n) = log(\frac{P(w_1, ..., w_n)}{P(w_1)...P(w_n)}) = log(\frac{f(w_1, ..., w_n) * N^{n-1}}{f(w_1)...f(w_n)})$$

PMI uses probabilities which can be estimated as $P(w) = \frac{f(w)}{N}$, where N is the total number of words, and $f(w)$ is the frequency of the word w. Its main problem is that it performs poorly with low frequency terms due to probability estimation error. This issue can be alleviated using Laplace smoothing [14] or using heuristic such as PMI^2 or PMI^3 which adds a second or third power to $f(w_1, ..., w_n)$ factor in the formula.

A conceptually similar measure called Dice factor exist and also suffer from the same low frequency problem. It was defined in [15] for the purposes of studying the amount of ecologic association between species. But is now frequently used in information retrieval (see [16], [17]).

T-score ([18],[19]), Z-score [20], and chi-square We can employ these measures to investigate the null hypothesis of independence. Independence test check that the probability that a set of words appear simultaneously is sufficiently higher than arbitrary chance. The issue here is that they are based on the Gaussian probability distribution, so we have to assume normality.

Jaccard Jaccard measures the ratio between how many times n elements co-occur and how many times each element occurs by themselves. Its formula is somewhat reminiscent of the PMI formula:

$$Jaccard(W_1, ..., W_n) = \frac{|W_1 \cap ... \cap W_n|}{|W_1 \cup ... \cup W_n|}$$

Such that W_i is the set of term that contains w_i.

Jaccard provides a measure between zero and one which is easy to interpret. It also does not take into account frequency which makes it a notable candidate for sparse term extraction.

2.2.2 Termhood

Termhood expresses how much a term is related to a specific domain. For example in financial corpus the term "financial assets" should have a higher termhood than "meeting room" since the former is more domain specific. Therefore termhood is also a crucial way of deciding the relevancy of a candidate term. They are four main measures that we can use to define termhood:

Frequency Indeed, frequency is intuitively the simplest answer to termhood. We can expect important and specific concepts to appear more frequently in a corpus about their specific domain. The issue with frequency is that it can pick up frequent expression that are not term at all but are nonetheless frequently used, for example: "for example". Such issue can often be greatly reduced with filtering and pre-processing text like removing stop-words or using TF-IDF measure to extract list of irrelevant words or sequence of words. But other issue cannot be solved with more filtering. For example the biggest problem we ran into was extracting part of terms instead of complete terms. To solve this, we need to use the C-value.

C-value First defined in [21], C-value extends the idea of frequency of a term t by taking into account the frequency other term h which contains t. This measure is useful to avoid picking up term sub-component which consequently must have equal or higher frequency than the term itself. Indeed the term "real time operating system" contains both the term "real time" and "operating system". Using a frequency measure the latter two term will have a frequency at least equal to their composite counterpart. C-value penalizes terms that are a subset of longer terms. This helps extract longer terms and avoid extracting sequences such as "time operating" which might be picked up by frequency ranking methods.

The C-value was also defined to be moderately but positively affected by longer term due to the logarithmic factor which again incentivizes the extraction of longer terms. But we drop this logarithmic factor in our implementation. Indeed since all 3-gram have the same size (number of non stop-words words) and we only compare n-grams for the same n, the logarithmic factor is not relevant.

The C-value is thus computed as follow :

$$C - value(t) = log_2(|t|) * (f(t) - \frac{1}{|H_t|} \sum_{h \in H_t} f(h))$$

Where t is the term being measured, H_t is the set of term longer than t and containing t, and $f(t)$ is the frequency of t.

NC-value C-value itself can be expended upon with the NC-value measure as defined in [22]. This measure uses contextual information extracted from term ranked by C-value. The reasoning behind NC-value is that there exists context words that tend to occur around terms. Detecting those context words can help us extract other terms.

Indeed, if a given word often precedes a term, we might want to look for other places where this word occurs and extracts the following sequence of words. The NC-value is thus defined as:

$$NC - value(t) = 0.8 * C - value(t) + 0.2 * CF(t)$$

$$CF(t) = \sum_{w \in C_t} \frac{f_t(w) * f_{wt}}{n_t}$$

Where t is a top-ranked term (by C-value), C_t is the set of context words of t, f_{wt} is the number of terms occurring with word w, n_t is the number of terms considered, and $f_t(w)$ is the frequency of w as a context word of t. We decided not to use NC-value in this thesis as it requires a more complex pipeline to compute but we did use the C-value.

SNC-value In [23], the TRUCKS model is presented, and it expands once more on the C/NC-value with its SNC-value. TRUCKS is a model which combines many sources of knowledge to extract terms. It uses an external thesaurus and a semantic network for terminological and semantic information. As well as syntactic, statistic, and contextual information through corpus analysis.

The system is constituted of three independent layers. The bottom layer operates a fully statistical base to compute the C-value. The inner layer combines linguistic and statistical information about context words to produce the context weight. The topmost layer derives an importance weight from syntactic, semantic, and terminological information about context words. The output of all these layers are combined to produce the SNC-value. The first and second layers thus only computes the NC-value explained above. But the third layer is where the innovative part of this paper lies.

Firstly, they use syntactic knowledge through the POS of boundary words to show that their type influence the likelihood of term being next to them. Boundary words are words that occurred immediately before or after a term and they may be a verb, an adjective, etc. They showed through experiments that verb and prepositions tend to co-occur more often with terms. This information is thus incorporated into the importance weight used to compute the SNC-value. Secondly, they consider the terminological information of context words. That is they assume context words which also represent terms are more likely to surround other terms. This information is also incorporated into the importance weight. Thirdly, semantic knowledge is provided by a semantic network. Which is employed to compute the similarity between terms. Indeed, they believe that a candidate term should be semantically similar to its context terms. This information is the last to be incorporated into the importance weight.

The importance weight is then added to the NC-value to derive the SNC-value of a term t :

$$SNC - value(t) = NC - value(t) + IW(t)$$
$$IW(t) = \sum_{w \in C_t} syn_t(w) + \sum_{h \in T_t} f_t(h) * sym_t(h)$$

Where C_t is the set of context words of t, $syn_t(w)$ is the syntactic weight of w as a context word of t, T_t is the set of context terms of t, $f_t(h)$ is the frequency of h as a context term of t, and $sim_t(h)$ is the similarity weight of h as a context term of t. Note that in their pipeline the candidate terms were already selected by the C-value. Thus the SNC-value is only there to perform a re-ranking of the terms extracted.

They evaluated their measure on a small corpus of eye pathology records pre-tagged with POS. Their experiments show that SNC-value performs better than NC-value. More precisely when ranking terms with the SNC-value higher up elements enjoy a grander chance to represent terms than if they were ranked using NC-value. Moreover, as we go down the list of term ranked with SNC-value the precision does not drop as much as with NC-value.

While this study is impressive and extremely insightful, the algorithm presented is relatively complex. We believe that simplicity, when sufficient, is always superior to more complex systems, that is we believe in minimal nescecary complexity. And while TRUCKS is clearly a powerful algorithm, simpler models are easier to maintain and implement and also tend to run faster. Moreover, we already discuss the problem with multi-stage systems and in this case the external thesaurus and semantic network used in this work adds another set of potentially compounding errors. Note that the study does not discuss whether or not long/complex terminology can be extracted using this method.

2.2.3 TF-IDF

TF-IDF is a well known measure in natural language processing. It measures the relevancy of a term by computing its frequency minus the inverse of its document frequency. The document frequency is defined as the number of documents containing this term. Formally we have:

$$TF - IDF_d(t) = f_d(t) * df(t)^{-1}$$

Where $f_d(t)$ is the frequency of t in d and $df(t)$ is the frequency of documents containing t.

The idea is that if a term appears a lot in a few documents, it is a specialized term. While if a term appears a lot in all documents, it will less relevant. For example, terms like "for example" will have an extremely low TF-IDF. While term such as "capital conservation buffer" will appear a lot in few documents and thus rank higher.

TF-IDF is typically used in information retrieval. That is, given a query, can we retrieve the relevant set of documents from a broader set of many more documents? Therefore, search engines remain the prime example of information retrieval application. In [24], they experimented with this measure on a collection of 1400 documents from the LDC's United Nations Parallel Text Corpus. They showed that indeed frequency alone didn't lead to relevant results for a given query. But when each term is weighted with its IDF to form the TF-IDF measure, the results are much more interesting.

Although its implementation is simple, the fundamental issue with this measure is computation time. Indeed computing the document frequency of each term can be extremely costly as we need to go through the whole set of documents. This is why we did not even test this measure although it can lead to good results. Moreover, [24] also noted that TF-IDF is unaware of synonyms or even plural form of words. This issue can be addressed with pre-processing such as lemmatisation or synonym detection.

2.3 Hybrid Approach

Hybrid approaches ordinarily employ linguistical technique as pre-processing step to detect stop-words or as a way to extract candidate terms using regular expression. While statistical measurements are used to filter and rank candidate terms.

An interesting study [2] solved the issue of complexity we discussed earlier with a simple trick. The question they asked is: what if we applied measures of unithood not only on sequences of words but also on their corresponding sequences of POS tags? Therefore, this allowed them to extract terms but also POS patterns for those terms without a human in the loop. Indeed, they were able to measure the termhood/unithood of POS pattern which means they were able to automatically extract syntactic patterns. They evaluated their method on the already POS-tagged Brown Corpus. This was done without domain expert, although this was not their choice

but rather a logistical limitation. They were able to achieve 85% precision on 3-grams but only 60% on 2-grams and the precision of 4-5-6-grams was below 50%. Therefore, their remarkably flexible approach can learn to extract terms from any languages which can be tagged using part of speech without any modification. One problem with this study is that the algorithm is extremely slow since the 250 000 words corpus was analyzed in 4h and 20 minutes. [2]. But the salient issue of this study is that once again the model is unable to effectively extract terms longer than three words.

In [3] they also used an hybrid approach allied with a linear conditional random field (lCRF) to extract multi-word terms. This study claims to be the first to use lCRF in the context of term extraction and the results are promising. Indeed, the lCRF stores many different patterns in the form of the various possible paths that can be traversed in the lCRF graph. Therefore, even small graphs can store an extensive number of possible patterns. Thus, using lCRF eliminates the problem of designing complex patterns by hands. Although humans are nevertheless required to craft the lCRF, it is considerably simpler than designing patterns directly. They experimented on part of a medical corpus MEDLARS. And they compared their results with a list manually crafted by a domain expert which contains term such as "toxicity of inorganic selenium salts." The results have shown that their model has a high recall of 84% but a precision just under 50%. The problem with this study is that training the lCRF requires annotated data which is always difficult and expensive to manufacture.

2.4 Limitations and challenges of past studies

As we have seen in this review of the literature, the problem of term extraction involves six fundamental challenges:

- **Evaluation.**
  Like many many studies have pointed out ([23],[7]), evaluating the quality of a term extracted is ordinarily an extremely subjective and difficult endeavour. It requires domain expert to validate the results which is slow and disagreement between experts can arise. This absence of objective evaluation makes it difficult to craft and compare models. Moreover, it is inconceivable to expect people to extract all possible terms from large documents. It is much more viable to ask them to decide on the termhood of extracted candidate terms. This means that computing recall is often not possible and only precision can be reasonably estimated.

- **Sparsity.**
  Some relevant term that we would hope to extract may not appear frequently and thus can be harder to detect. This problem can be solved by using measure that are unaffected by frequency such as Jaccard. In [25], corpus information sparseness is dealt with using a weakly supervised approach combined with the high redundancy of the web. But frequency is still an important measure and thus we somehow need to mix it with a non-frequency-based measure to get the best of both worlds.

- **Long terms.**
  The complexity of current methods such as linguistical approach can quickly get out of hand when trying to extract long terms and the results are often quite noisy. It is clear

[2]For comparison, in the results section of this thesis, we will see that our algorithm analyzes about 5,000 words a second. Thus 250 000 would be analyzed in under a minute. And although our machine is notably faster it likely not 260 times faster.

that we cannot create indefinitely more complex regular expression without inevitably striking a concrete wall of unmanageable complexity. We require novel technique capable of detecting such long term in a more straightforward and automated manner. Hybrid methods such as [3] and [2] shows that there is a possibility to automate the creation of syntactic patterns. But these studies were not capable of extracting long terms. Especially term as long as: "financial assets designated at fair value through profit or loss".

- **Frequent yet irrelevant constructs.**
  Many constructs such as "the information specified in" or "for example" can appear frequently while being irrelevant. This issue can be reduced to some degree by using better filtering and pre-processing technique on the data. Indeed linguistic and statistical methods can both provide techniques for detecting such constructs. Whether it is through syntactic parsing or termhood measurements.

- **Lack of domain specific knowledge.**
  High-level domains such as European financial regulation provide little secondary sources of domain knowledge available for us. This means we frequently need to develop methods from scratch and with little support to confirm our result. This slows down the modelling process significantly. The solution to this is more structured and available data. But although a lot of data is produced every day, most of it is not annotated. Indeed manually annotating data is a time-sink, thus in the meantime it is preferred to utilize unsupervised methods or minimally supervised methods for terminology extraction.

3 Methodology

In this section we will explain the complete methodology we developed for extracting complex terminology. We will first discuss the how data is pre-processed using a variety of operations aimed at simplifying the analysis. We will subsequently discuss the underlying principles surrounding our method such as the challenges we faced, the definition of supergrams, our custom noise (SWF) and term relevancy measure (CPS^2), as well as the dynamic frequency we used. Next, a comprehensive explanation of the algorithm from extraction to ranking will be given. And finally, we will focus on more minor yet interesting details such as the way measures were normalized, how memoïsation was applied, and we will present our abbreviation extraction system which is a secondary contribution.

3.1 Pre-processing

Preparation The corpus used was crawled from the web using links [3] provided to us by DG FISMA, the financial branch of the european union. And the documents we gathered were pre-processed by removing all HTML tags.

Although HTML tags can often be of great value for text analysis. We decided they were irrelevant in our case due to their structure which was unhelpful. Hence, ultimately our term extraction algorithm takes raw text as input.

Filtering After being pre-processed by our data crawler, the input data is first filtered. The goal of filtering is to decrease noise by reducing the uninformative complexity of a given dataset. In NLP the removal of stop-words is typically the first step as they do not contain any information.

But in our case we decided to keep them. The main reason being that we want to extract terms and their stop-words to have a human-readable output. But also because we can use stop-words as a way to measure noise in our results, indeed the absence of information is itself a kind of information.

In our algorithm, filtering involves lowering all upper-case letter, fixing the contractions (e.g. "I'll" is changed to "I will"), and removing/replacing some elements. For example, a space has been added after every full stop (`"."` became `". "`) to avoid tokenization mistake. And new line elements such as `"\r"` or `"\n"` were also replaced by `" | "` to avoid extracting elements separated by new lines.

Tokenization Tokenization is the process of transforming a sequence of characters into a sequence of words and it is the last pre-processing step. This is primarily performed by exploiting spaces to detect the words and extract them. In our implementation though we have augmented traditional tokenization by forcing words surrounded by parentheses to collapse into one token. For example after traditional tokenization, we may obtain the sequence ["delegated", "regulation","(","eu",")"], but after our tokenization we will have: ["delegated", "regulation","(eu)"]. This is implemented to ensure that isolated elements surrounded by parentheses are extracted as one element with their parenthesis, such elements will then be considered stop-words. We have also made sure that words linked by hyphens are extracted as single token. For example,

[3] Example: `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?qid=1524570109266&uri=CELEX:02014R0680-20161201&from=EN/`

"off-balance-sheet" would normally be tokenized as ["off" "-" "balance" "-" "sheet"] but in our implementation it is extracted as a single token.

3.2 Term extraction : principles

Challenges Five main challenges lead to the development of our novel algorithm:

First, as stated above, we wish to extract long terms for which classical methods are not well suited (See [1] or [10]). In our case, a particular long term of interest is:

"financial assets designated at fair value through profit or loss".

Secondly, as often in data science, real life is less convenient than academic exercises. Some studies make use of weakly, semi, or fully supervised learning such as [3]. But annotating data is slow, difficult and requires costly domain expert and sometimes it is logistically unviable (As in [2]). Therefore the term extraction algorithm we developed is unsupervised and only the evaluation will be done by domain experts.

Thirdly, considering the complexity of financial regulatory terms, we cannot use part-of-speech tagging allied with pattern seeking algorithms to extract terms efficiently. This is true especially for long term such as "financial assets designated at fair value through profit or loss" which are overly complex for POS-based method to be viable. Indeed, we would need an overly complex set of patterns to extract such terminology efficiently. This is not advisable nor would it be maintainable or flexible, and it would also probably collect a lot of noise along the way.

Fourthly, it is often the case in NLP that we can use external support to analyze texts such as ontologies. But the domain in which this work is based on (financial regulatory text) is not supported by external ressources. Thus, we need to create an algorithm capable of working from scratch with no knowledge about the domain.

Lastly, we need to rank our candidate term efficiently. To achieve this, we will introduce a new measure for ranking terms based on a mix of termhood, unithood, and noise measures. We designed this measure to extract complete terms and avoid the extraction of sub-set of terms or irrelevant constructs.

A statistical approach Our algorithm for term extraction is purely based on a statistical approach. More precisely, the algorithm extracts n-grams and what we call supergrams and then performs measurements to help filter what should be considered as terms. An n-gram is a contiguous sequence of n co-occurring words/tokens while a supergram is a contiguous sequence of any number of co-occurring n-gram of any n.

The choice of basing our algorithm solely on a statistical approach was deliberate. We believe that simpler models are more appropriate and more elegant. And we have already discussed that linguistic methods lead to complex multi-stage models in which errors at each stage compound on each other. While some complex model perform well [23], if a simpler model performs just as well or better it is preferable. Moreover, the simpler the model the easier it is to understand and maintain.

Supergrams As we have already discussed, financial regulatory text contains a complex terminology. Indeed, in such technical domains terms can be made of 10 words or more. Thus, we need to be able to extract them.

We wish to extract long terms, that is terms containing four or more non stop-words words. But extracting n-gram becomes computationally expensive as n increases. Thus, extracting long terms using n-gram mining is slow and inefficient.

Also, as n increases the extracted n-grams are less relevant and more noisy since long terms are relatively rare. This leads to many useless and expensive computations for few usable results. This is why we decided to extract 2,3,4-grams only, because in our context, extracting 5-grams and above does not lead to many relevant results.

Finally long terms can often be broken down into sub-terms. Therefore, we define a supergram as a contiguous valid sequence of n-gram for any n. The validity of a sequence is defined in the supergrams extraction section. This will help us find longer terms made of many smaller ones. An example of an extracted supergram would be: "financial assets designated at fair value through profit or loss".

Proposed noise measure, SWF A typical issue in data science project is dealing with noise and noisy results. It is often challenging to come up with an adequate measure for noise but in our case we managed to do so. Indeed, most NLP application get rid of stop-words in the pre-processing step, but we don't. One because the output is expected to be human readable. And two, this offers us an advantage for constructing a noise measure.

We call this measure stop-words fraction or SWF. It measures the ratio between the number of stop-words with the total number of words in a sequence. For example, "financial assets designated at fair value through profit or loss" contains 10 words in total and 3 of them are stop-words. Thus, this term has a noise level of 30%.

More formally, we define SWF as:

$$SWF = \frac{\#stop - words}{\#words}$$

Using this measure will allow us to easily discard noisy candidates as we will see in the results of the experiments.

Proposed term measure, CPS^2 In the literature section we have shown two main types of measure: termhood and unithood. And now we have just introduced our noise measure. Therefore, we can now introduce our proposed measure for term extraction, called CPS^2.

The idea is that if we mix a termhood, a unithood, and a noise measure we can craft a precise and reliable metric for ranking candidate terms. We have chosen the C-value, PMI, and SWF to craft this measure (See 2.2), it is defined as follow:

$$CPS^2 = C - value * PMI * (1 - SWF)^2$$

Notice the noise measure has been square to give more emphasis to term with few stop-words. To score high with this measure, a term will need three conditions. First, it needs to be frequent enough while not being a subset of too many other frequent terms we have extracted, this is measured with C-value. Second, the elements that makes up the term need to co-occur

with each other significantly, this is measured with PMI. Thirdly, the term needs to contain a minimal amount of stop-words to avoid noisy results, this is measured with SWF.

In the experiments results section, we will show that this measure outperforms all other tested measure at extracting terms in our domain.

Dynamic frequency threshold The least frequent an element is, the noisier it tends to be. Therefore, we need a frequency threshold for n-grams and supergrams extraction to filter out unlikely candidate. But as a document increase in size, the frequency of noisy element increases as well. Hence, we need to define a threshold that changes with document size.

We used $\frac{docSize}{30000} + 2$ as our threshold. It maintains a minimum of two because we want to make sure we never extract elements that appear only once. And for every 30 000 words in a document, it gains a unit. Thus, any candidate term needs to be at least equal to the dynamic threshold to be extracted.

The way this particular threshold was chosen depends on our corpus. We look at the output of documents of various lengths and tried to determine under which frequency the number of correctly extracted terms, or lack thereof, becomes undesirable. It was self-evident from our experiments and little knowledge that below some frequency threshold, most of the output was noise. And that this threshold changed with the size of the document. Thus, we chose 30 000 through a subjective analysis.

3.3 Term extraction : implementation

Now that we have laid out the principles of our methodology, we may discuss the implementation of the algorithm. To begin with, we will look at the highest level of abstraction to understand the full pipeline of term extraction 1. Next, we will discuss n-grams and supergrams extraction. At long last, we will discuss how candidate terms are selected.

3.3.1 The full algorithm

Figure 1: Complete pipeline

First we can briefly have a high-level view of the algorithm 1 before diving into the details in the following sections. The algorithm starts by extracting n-grams (n=2,3,4) and filtering them

using multiple conditions. It subsequently uses the extracted n-grams to derive supergrams via overcomposition. Those supergrams are also filtered with a variety of techniques. We can then extract potential terms by applying our CPS^2 measure to the n-grams to rank them before outputting a list for each n. Finally, supergrams are also ranked, this time by their frequency, before outputting them as a list of potential long term.

3.3.2 N-grams extraction

Figure 2: N-gram extraction

Shift register idea The idea used to extract n-gram is similar to how a shift register works in electronics. That is to extract n-gram, a window of n non-stop-words elements (see 3) slides over the text and detects every n-gram. This window W is also associated with a vector of stop-word vector S such that a term is a sequence of words each separated by a sequence of stop-words:

$$W_1 S_{11} ... S_{1i} ... S_{1n} W_2 ... W_i S_{i1} ... S_{ii} ... S_{in} W_{i+1} W_n$$

Note that S_n is not used to derive the extracted n-gram since we are not interested in the stop-words following the last word of the n-gram.

Every time we read a new token from the document being analyzed, if it is a stop-words it is simply appended to the last vector of S. But if it is an actual word, the window W and stop-word vector S is shifted to the left. This is done by dropping the first element of W and first vector of S. Only then can we add the newly found word at the end of W.

For example, lets extract a few 3-grams from the sentence: "Encumbered assets or collateral received and other off-balance-sheet items may be pledged to secure funding.". It works in the same way for 2/4-grams. Thus, the window W starts with the first three non-stop-words words and W = ["Encumbered", "assets" ,"collateral"]. The stop-word vector S, therefore, contains [[],["or"],[]] since there is the stop-word sequence ["or"] between "assets" and "collateral" but there is none between "Encumbered" and "assets" or after "collateral". The first 3-gram we extract is consequently "Encumbered assets or collateral".

As we encounter a new token "received", we shift W and S to the left. This is done by dropping the first element of W and S and then adding "received" to the end of W and any stop-words sequence following "received" to the end of S, in this case ["and","other"]. We now have: W=["assets" ,"collateral","received"] and S=[["or"],[],["and","other"]]. A new 3-gram is then extracted "assets or collateral received", note that the last element of S is never used to derive an n-gram.

As we continue to a new token and shift W and S, we have W=["collateral","received","off-balance-sheet"] and S=[[],["and","other"],[]] And again a new 3-gram is extracted "collateral

received and other off-balance-sheet". Note that here we demonstrate an example of our augmented tokenization where "off-balance-sheet" is extracted as a single token instead of five. Additionally note that S is indeed a vector of vectors since its second element here is ["and","other"] which is the stop-words sequence found between "received" and "off-balance-sheet".

Figure 3: Visual demonstration of the interlaced shift register idea. Red boxes represent the content of W while white boxes represent the content of S's vectors.

Term validity Every time we shift our window, we encounter a new possible candidate term. After fusing the element of W and S together to derive the newly detected term we can test if it is valid. A term is valid if it respects four conditions:

1. It does not contain numbers.
2. It does not contain symbols/punctuation. With an exception for isolated word between parentheses such as "(eu)" which are classed as stop-words.
3. It does not contain tokens that are made of only one character. This is to prevent any alphabetical punctuation like "a)" which would be tokenized as ["a",")"].
4. It does not contain invalid words. For this we compiled a list of invalid words tailored for the corpus studied. It contains many symbols and irrelevant constructs like "iv", "subparagraph", "ten", etc.

3.3.3 Supergrams extraction

Extraction To extract supergrams, we first need a list of all extracted n-grams occurrence sorted by positions. Such a list allows us to iteratively and quickly derive supergrams. For example we might have the list in table 1, in which we can notice there are three 2-grams and a 3-gram.

A supergram is a sequence of validly linked n-grams. Two n-grams are validly linked if they are only separated by a sequence of stop-words and that none of these stop-words belongs to the list of invalid words. Note that this sequence of stop-words can be empty. Additionally note that many n-grams overlap each other, this means that one needs to remove overlapping elements from successive n-grams before deriving a supergram.

For example, using the list in table 1, we can extract "financial assets designated at fair value through profit or loss". In this case "financial assets" and "assets designated at fair" overlap

Figure 4: Supergram extraction

...
financial assets
assets designated at fair
fair value
profit or loss
...

Table 1: This is a sample list of all n-grams (n=2,3,4) occurrences extracted from a document and sorted by their positions.

with the word "assets". This is also the case for "assets designated at fair" and "fair value", they overlap with the word "fair". And finally, "fair value" and "profit or loss" do not overlap and are linked by the stop-words sequence ["through"]. From this we can see we can indeed re-create the term "financial assets designated at fair value through profit or loss". We could equally have extracted the n-grams "financial assets" and "designated at fair" which are linked by an empty sequence [].

After extracting all valid sequences of n-grams, we can filter the resulting supergrams.

Filtering We need to filter supergrams to get rid of irrelevant results. First, if a supergram is too small it might be already extracted via n-grams extraction. Since we do not want both system to compete, supergrams need to at least contain 4 non-stop-words words. This does mean that supergrams compete with 4-grams but 4-gram tend to be noisier anyway as we approach the limit of efficient extraction with the n-grams approach. Remember we discuss that as n increases, the number of relevant candidate decreases and thus noise starts to fill the gaps. Thus, we can still extract relevant supergrams that would not be highly ranked in the ultimate 4-gram output due to the abondance of noise. Therefore, we assume supergrams are less noisy by their construction, that is overcomposition of potential terms. Second, infrequent supergrams are removed using the dynamic frequency threshold we presented. Third, a maximum size of 10 non-stop-words words has been decided empirically for supergrams. Any supergram that is longer will be pruned to this maximal size by dismissing the extra word at the end of the supergram. Finally, any supergram that is a subset of any other is removed.

3.3.4 Selecting results

Finally, let's look at how candidate terms are selected for both supergrams and n-grams.

Supergram Supergrams are sorted by frequency with no additional filtering. Thus, the whole list is outputted. This is because we assume that any supergram extracted is a good candidate term. This is because we believe the filtering and constraints applied during supergrams cleaning is enough. And that by contruction (overcomposition of smaller terms) supergrams are more likely to be terms. The results of our experiments will demonstrate that.

N-grams Extracting n-grams is a bit more complex. To begin with, we get a list of all the supergrams and n-grams that we extracted. Next, any n-gram which frequency is lower than the dynamic threshold is removed. And any n-gram that is a subset of any supergram is removed. At long last, the n-grams are ranked with our CPS^2 measure.

3.4 Implementation decision for the different measures

Measures like frequency, DICE, and PMI were normalized by dividing them by their maximal value. More precisely, for a given k, we computed the PMI score for all k-grams and used the maximum PMI computed to normalize the PMI score of all of the k-grams. We apply the same treatment for DICE and frequency normalization.

The reason for normalization is two-fold. First, in our experiments it is easier to compare measure that are on a similar scale. Second, we want to make sure that in our CPS^2 measure, PMI, C-value, and SWF possess the same relative weight.

About C-value, remember that for a term t it is computed using a set of longer terms that contains t.

$$C - value(t) = (count(t) - \frac{1}{|H_t|} \sum_{h \in H_t} count(h))$$

Therefore, we have to select a set of candidate terms H_t containing longer elements than t and which contains t. To compute the C-value of an n-gram, we need to compute H_t. In our case, H_t is made of the top-20 m-gram for each m (m>n) extracted and sorted using frequency and which contains the term being measured.

3.5 Abbreviation extraction

As an ancillary contribution, we also developed an algorithm which extracts acronyms and their corresponding extended form. To achieve this, we assume people tend to define abbreviations by writing the extended form before the abbreviation, for example: "Internal assessment approach (IAA)". This assumption makes sens since this is how we traditionally defined acronyms.

To achieve this, we simply extracted all tokens primarily made of upper-case letters (>50% of upper-case). And everywhere those tokens appear we look at the preceding non stop-words words to see if their initials correspond to the upper-case character of the potential acronym. We only compare upper-case in the abbreviation with the initial of the preceding word sequence, thus we ignore punctuation and stop-words. Therefore, the abbreviation "BeNeLux" can be extracted if preceded by the sequence "Belgium, Netherlands, and Luxembourg". Thus, with such system we can extract terms like "INTERNAL ASSESSMENT APPROACH" and their abbreviation "IAA".

3.6 Memoisation

Without any optimization, the algorithm is extremely slow, it can take hours to analyze small texts. Hence, we required a way to force the algorithm to run faster, for that we made extensive use of memoïsation. The idea is that if an intermediate value is hard to compute and/or is needed frequently, we might want to preserve it in a cache to avoid re-computation. Memoïsation considerably improved performance to an impressive degree; the time required to analyze text was divided by at least 200. Memoïsation is used for caching the various measure employed on n-grams, the output list of supergram, the tokens of n-grams and other important intermediary functions.

For example the frequency of an n-gram is frequently used to compute its C-value and the one of other k-gram containing it (k<n). To compute the frequency of an n-gram we need to go through the whole input which means that we expect a complexity of O(N) for this task, where N is the size of the input. Considering the many calls to the functions which computes the frequency, without memoïsation we would get a complexity of O(CN), where C is the number of calls. While with memoïsation we have a complexity of O(N) on the first call and O(1) for the following calls. This means the complexity of computating C calls of the frequency function is O(N+C) which leads to a radically faster computation time.

4 Experiments and results

This section will discuss the results of the experiments performed for this project. All term extraction experiments were performed on three big documents and the extracted terms displayed in the various tables come from the same document [4]. The algorithm we are presenting in this thesis was parameterized in such a way to extract up to 4-grams. This is critical for computing measure like C-value which depends on a set of longer candidate terms than the one being measured. Therefore, we will exclusively show the resulting 3-grams for the different n-gram-based experiments.

In the results displayed on various tables, terms will be classified in three classes. If an element is a term, it will be assigned to the class Y. If an element is not a term, it will be assigned to the class N. If an element is part of a term, it will be assigned to the class P.

About the performance measure, we exclusively used precision since we did not have access to a list of all correct terms. Indeed, to compute the recall one needs a list of all terms to be extracted. This was not possible because domain experts did not have the time for such job. On the other hand, to compute precision, we merely require to classify the extracted terms. The classification was performed by ourselves, although we asked domain experts at the DG FISMA, they were busy.

4.1 Data

In this work, the data employed to extract terms is a subset of the EUR-lex corpus. This corpus is made of financial regulatory documents maintained by the financial branch of the European commission, DG FISMA. In total 542 documents were gathered and pre-processed. Each text contains between 140 and 242 000 words although they are only three big documents the rest has less than 140,000 words. Figure 5 gives a more detailed view of the size distribution of the documents (the y-axis is a log scale.).

About the content of these documents, the most frequent word in the corpus is "the" with a frequency of 533. And this is what one should expected in most English corpus. The average frequency of words is 10 which is smaller than we expected.

Figure 6, describe the distribution of the words (the y-axis is a log scale.). As we can see this is what one would expect, words are distributed in a ziphy fashion. That is, on a log scale it appears as though we have a linear relation between occurrence and diversity of words. Few words are employed most of the times while many words are rarely used. The tallest and first bin is interesting as it disturbs the linear relationship.

This corpus itself is made of financial regulatory text. This means that they are professionally drafted with virtually no grammar or spelling mistake and structured in a similar manner. Indeed, we will briefly discuss that one of the challenges was the extraction of long terms. This is only possible because legislative text tend to be very precise and thus complex concepts are referred to using the same exact words in different places. Consequently, the complexity and noise of such text is supposedly lower than other corpus used in NLP analysis such as twitter tweets; at least for the problem of term extraction.

[4] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?qid=1524570109266uri=CELEX:02014R0680-20161201from=EN/

Figure 5: Distribution of the size of the documents in the corpus provided by DG FISMA

4.2 Demonstrating POS tagging inefficiencies

Our first experiment will show the limitations of seeking patterns using part-of-speech tagging. Table 2 lists 25 terms extracted using POS tagging with pattern $[A^+N^*]$, with A for adjectives and N for nouns. Each term is associated with a frequency and a class manually assigned and validated by ourselves. Specifically, members of DG FISMA who deal with the Eur-lex corpus helped us validate our results.

As we can expect from a state-of-the-art method, most of the results extracted with POS tagging are correct. Indeed, the extracted term contains 21 correct answers, 1 partially correct answer and 3 incorrect ones. This gives us a precision of at least $\frac{21}{25} = 84\%$ if we only count the Y's.

But remember that we only extract term of the form $[A^+N^*]$. If we wish to extract other kinds of term, we need to design other patterns. As we discussed before this is the main limitations of pattern-seeking POS-based methods. Depending on the application it will be either impossible or cumbersome to design all possible patterns to extract terms. Or else we run the risk of missing many terms.

For example here are three long terms extracted with our algorithm:

"financial assets designated at fair value through profit or loss".
"extremely high quality covered bonds".
"transferable assets representing claims".

Figure 6: Distribution of the occurrence of the words in the corpus provided by DG FISMA

If we apply POS tagging, here is what we get:

ADJ NOUN VERB ADP ADJ NOUN ADP NOUN CCONJ NOUN
ADV ADJ NOUN VERB NOUN
ADJ NOUN VERB NOUN

As we can see the basic pattern $[A^+N^*]$ is not capable of extracting such terms. Clearly, we could create a disjunction of the three patterns such as [pattern1 or pattern2 or pattern3]. But when designing patterns we would prefer them to be able to generalize and thus one cannot simply hardcode disjunction of isolated examples. Designing patterns which generalize well and extract terms correctly becomes more and more tedious as we try to extract longer and more complex terms. This is why POS-based methods are limited in their ability to extract complex terminology.

Our algorithm, on the other hand, extracts these term in an automated manner. Indeed, there is no need for domain expert to help in the creation of complex patterns. Hence, it is evident that although we have shown in table 2 that POS tagging remains an excellent method for extracting simple terminology. We argue that it is not reasonable to try and generalize such techniques for the extraction of longer terms. And anyone who wishes to employ such techniques to extract long and complex terms will hit a wall of complexity when trying to design efficient patterns.

While it is not an impossible task, our algorithm proposes a considerably easier alternative.

Table 2: Element extracted using POS tagging with pattern $[A^+N^*]$

POS : term extracted		
N	359	Annex V.Part
Y	234	credit quality step
P	218	fair value
Y	198	Debt securities
Y	173	financial assets
Y	172	central banks
Y	166	reporting institution
Y	148	financial sector entities
N	144	BAD art
Y	142	significant investment
Y	138	liquid assets
Y	132	balance sheet items
Y	115	Other financial corporations
Y	115	Non - financial corporations
Y	112	credit institutions
Y	110	credit risk
Y	109	defaulted exposures
Y	107	balance sheet
Y	106	General governments
Y	104	Central banks
Y	104	Equity instruments
Y	102	own funds requirements
N	96	Total amount
Y	95	Retail deposits
Y	95	Unsecured wholesale deposits
21Y 3N 1P		

4.3 Comparing measures

We will now compare the various measures presented in our review of the literature on statistical methods. From these experiments we will explain how and why we adopted our CPS^2 measure. For that, three types of measures need to be tested: termhood, unithood, and noise. Finally, our CPS^2 measure will also be tested.

4.3.1 Termhood : Frequency and C-value

Termhood is the measure of how specialized a term is for a given document. We will experiment on two measure of termhood: frequency and C-value. In the table 3, we compared both measures. Each term is associated with its a score yielded by the measure being evaluated and a class manually assigned and validated by ourselves.

The precision achieved when using frequency is 70% on average while C-value has a precision of 75% on average. As we expected the C-value is a more adequate measure for extracting complete terms but not as much as we hoped. This is due to the heavy filtering, and preprocessing that occurs before the n-grams are assigned a score.

Therefore, as we expected, we choose the C-value as our termhood measure but it is arguable whether the performance cost is worth the precision gain. Indeed, computing the C-value of a term is relatively a much more demanding task than simply computing its frequency.

Table 3: 3-grams extracted using improved c-value and normalized frequency

IC-value			Normalized frequency		
N	1.0	delegated regulation (eu)	N	1.0	delegated regulation (eu)
Y	0.69	unsecured wholesale deposits	Y	0.69	unsecured wholesale deposits
Y	0.54	deposit guarantee scheme	Y	0.54	deposit guarantee scheme
Y	0.44	legal references and instructions	Y	0.44	legal references and instructions
N	0.37	exempted from the cap on inflows	N	0.37	exempted from the cap on inflows
P	0.26	gross holdings included	P	0.26	gross holdings included
Y	0.26	own debt securities	P	0.26	representing claims on or claims
Y	0.25	reporting reference date	P	0.26	claims on or claims guaranteed
Y	0.25	balance sheet exposures	Y	0.26	own debt securities
Y	0.23	multilateral development banks	Y	0.25	reporting reference date
Y	0.22	own covered bonds	Y	0.25	balance sheet exposures
Y	0.21	available-for-sale financial assets	Y	0.23	multilateral development banks
P	0.21	for-sale financial assets	Y	0.22	own covered bonds
Y	0.21	extremely high liquidity	Y	0.21	available-for-sale financial assets
Y	0.2	own estimates of lgd	P	0.21	for-sale financial assets
Y	0.2	statement of profit or loss	Y	0.21	extremely high liquidity
P	0.17	held by the reporting institution	Y	0.2	own estimates of lgd
Y	0.17	financial guarantees and other commitments	Y	0.2	statement of profit or loss
N	0.17	underlying assets in point	Y	0.17	direct gross holdings
N	0.17	lent and the following collateral	Y	0.17	off-balance sheet exposures
Y	0.17	groups of connected clients	P	0.17	held by the reporting institution
Y	0.17	joint ventures and associates	Y	0.17	financial guarantees and other commitments
Y	0.16	deferred tax liabilities	N	0.17	underlying assets in point
P	0.16	non-central government public	N	0.17	lent and the following collateral
Y	0.15	organized market options	N	0.17	cap on inflows as specified
Y	0.15	group of connected clients	Y	0.17	groups of connected clients
P	0.15	classified as operational deposits	Y	0.17	joint ventures and associates
Y	0.14	liquid assets template	Y	0.16	deferred tax liabilities
P	0.14	financial liabilities measured	P	0.16	non-central government public
P	0.14	domestic currency of the central	Y	0.15	organized market options
19Y 7P 4N			19Y 6P 5N		

4.3.2 Unithood : Jaccard, PMI, Dice, and T-score

Unithood is a measure of how much a sequence of words acts as a distinct unit. We will experiment and compare four of those measures: Jaccard, PMI, Dice, and T-score. In the table 4 and 5 we can observe the results of the experiments on these measures. Once more, each term is associated with its a score yielded by the measure being evaluated and a class manually assigned and validated by ourselves.

While T-score performed the worst with about 75% of precision on average, using PMI lead to the best results with about 80% precision on average (both average are over the three documents). Our small sample size (because of the fact that we had to annotate manually the extracted elements) means that the difference between T-score and PMI might not be statistically significant. What sets PMI apart from all the other measure is that it does not converge so rapidly to zero. Indeed, this represents a fundamental problem with Jaccard and to a lesser extent Dice and T-score. We therefore choose PMI as our measure of unithood although the other measure did not significantly under-perform.

Table 4: 3-grams extracted using T-score and normalized Dice

T-score			Normalized Dice		
N	1.0	delegated regulation (eu)	Y	0.66	joint ventures and associates
Y	0.83	unsecured wholesale deposits	Y	0.58	unsecured wholesale deposits
Y	0.73	deposit guarantee scheme	Y	0.46	groups of connected clients
Y	0.66	legal references and instructions	N	0.42	delegated regulation (eu)
N	0.61	exempted from the cap on inflows	Y	0.38	obligor grades or pools
P	0.51	representing claims on or claims	Y	0.37	legal references and instructions
P	0.51	gross holdings included	Y	0.34	multilateral development banks
P	0.51	claims on or claims guaranteed	P	0.31	obtained by taking possession
Y	0.51	own debt securities	P	0.3	representing claims on or claims
Y	0.5	reporting reference date	Y	0.28	deposit guarantee scheme
Y	0.5	balance sheet exposures	P	0.28	claims on or claims guaranteed
Y	0.48	multilateral development banks	P	0.28	gross holdings included
Y	0.47	own covered bonds	P	0.26	non-central government public
Y	0.46	extremely high liquidity	Y	0.22	obligor grade or pool
Y	0.46	available-for-sale financial assets	Y	0.22	master netting agreement
P	0.46	for-sale financial assets	Y	0.21	group of connected clients
Y	0.44	own estimates of lgd	Y	0.19	extremely high liquidity
Y	0.44	statement of profit or loss	Y	0.19	organized market options
Y	0.42	direct gross holdings	Y	0.18	defined benefit obligations
N	0.42	lent and the following collateral	P	0.17	claims guaranteed by sovereigns
Y	0.42	financial guarantees and other commitments	Y	0.17	defined benefit pension
N	0.42	underlying assets in point	N	0.16	exempted from the cap on inflows
N	0.42	cap on inflows as specified	Y	0.15	nature of the immediate counterparty
Y	0.42	off-balance sheet exposures	Y	0.15	direct gross holdings
P	0.42	held by the reporting institution	Y	0.14	defined benefit plan
Y	0.41	joint ventures and associates	Y	0.13	long settlement transactions
Y	0.41	groups of connected clients	Y	0.13	deferred tax liabilities
P	0.4	non-central government public	Y	0.13	exempted ccp leg
Y	0.4	deferred tax liabilities	P	0.12	classified as operational deposits
Y	0.39	organized market options	Y	0.12	statement of profit or loss
19Y 6P 5N			21Y 7P 2N		

4.3.3 Noise : ISW

Our third measurement for ranking relevant term is noise. Being able to measure noise is generally difficult but whenever possible it can lead to striking results. In our algorithm we extract terms along with their stop-words. Thus, we can use the number of stop-words as a measure of noise. Remember, we define our noise measure as $SWF = \frac{\#stopwords}{\#words}$.

This measure allows us to remove most of the noise, indeed it is our best measure so far. In table 6 the top 30 elements extracted with this measure were all devoid of stop-words and the results is a precision of 84% on average. Although it is a reliable measure, it does not convey much information on it's own, indeed all of the term extracted here have the same score. This is because many extracted n-gram do not contain stop-words and this measure only penalize those that do. In the following section we will see how this noise measure helps our CPS^2 measure produce highly precise results.

4.3.4 Extracting terms using our CPS^2 measure

Finally in table 7, we will present the results of our proposed measure, CPS^2. This measure is constituted of a mix of termhood, unithood, and noise measures. More precisely, $CPS^2 = IC - value * PMI * (1 - SWF)^2$ Squaring the noise measure ensures that if a term contains

Table 5: 3-grams extracted using Jaccard and normalized PMI

Normalized PMI			Jaccard		
Y	0.77	joint ventures and associates	N	0.76	delegated regulation (eu)
Y	0.77	obligor grades or pools	Y	0.34	joint ventures and associates
	0.77	obtained by taking possession	Y	0.3	unsecured wholesale deposits
Y	0.75	nature of the immediate counterparty	Y	0.17	groups of connected clients
Y	0.74	groups of connected clients	Y	0.13	obligor grades or pools
	0.73	non-central government public	Y	0.12	legal references and instructions
Y	0.73	master netting agreement	Y	0.12	multilateral development banks
Y	0.73	multilateral development banks	P	0.1	representing claims on or claims
Y	0.72	defined benefit pension	P	0.09	direct gross holdings
Y	0.71	organized market options	P	0.09	gross holdings included
Y	0.71	obligor grade or pool	Y	0.09	deposit guarantee scheme
Y	0.7	defined benefit obligations	P	0.09	claims on or claims guaranteed
Y	0.7	unsecured wholesale deposits	Y	0.08	organized market options
Y	0.7	group of connected clients	Y	0.08	obligor grade or pool
Y	0.7	exempted ccp leg	Y	0.08	nature of the immediate counterparty
Y	0.7	defined benefit plan	P	0.07	obtained by taking possession
Y	0.68	own estimates of lgd	P	0.07	non-central government public
P	0.68	representing claims on or claims	Y	0.07	group of connected clients
Y	0.68	deferred tax liabilities	N	0.07	exempted from the cap on inflows
Y	0.68	claims guaranteed by sovereigns	Y	0.06	master netting agreement
P	0.67	gross holdings included	Y	0.05	defined benefit obligations
Y	0.67	european financial stability	Y	0.05	long settlement transactions
P	0.67	claims on or claims guaranteed	Y	0.05	extremely high liquidity
Y	0.67	legal references and instructions	P	0.05	classified as operational deposits
Y	0.67	extremely high liquidity	Y	0.04	defined benefit pension
P	0.66	domestic currency of the central	P	0.04	new funding obtained
N	0.66	new funding obtained	Y	0.04	defined benefit plan
Y	0.66	long settlement transactions	Y	0.04	breakdown of loans and advances
N	0.66	reported in the following subcategories	Y	0.04	claims guaranteed by sovereigns
N	0.65	delegated regulation (eu)	Y	0.03	deferred tax liabilities
23Y 4P 3N			20Y 8P 2N		

stop-words, it must have a highly significant termhood and unithood to be selected.

With an average precision of 89% over the three documents tested, our CPS^2 measure leads to state-of-the-art performance. The sole issue with it is it's tendency to go quickly towards zero. We can see that terms such as "groups of connected clients" are selected even-though they may contain stop-words. This means that the unithood and termhood of this sequence is high enough to overcome the penalizing effect of our noise measure. Thus the measure acts as intended and leads to extremely precise and reliable results over the 3 documents we tested.

4.4 Supergrams

As we discussed before our algorithm strength is in its ability to extract long terms. These so-called supergrams are extracted by linking together sequences of n-grams that are validly linked. We believe these results represent the state-of-the-art in long term extraction. On average over the three document we tested, we have a precision of over 90%. In table 8 we can examine a sample of the results.

Table 6: 3-grams extracted using our noise measure

1-SWF		
Y	1.0	reporting reference date
Y	1.0	traded debt instruments
Y	1.0	deferred tax liabilities
Y	1.0	defined benefit pension
Y	1.0	multilateral development banks
Y	1.0	direct gross holdings
P	1.0	gross holdings included
Y	1.0	consolidated own funds
Y	1.0	balance sheet exposures
Y	1.0	long settlement transactions
Y	1.0	off-balance sheet exposures
N	1.0	delegated regulation (eu)
Y	1.0	available-for-sale financial assets
Y	1.0	transferred financial assets
Y	1.0	net defined benefit
Y	1.0	defined benefit plan
P	1.0	financial liabilities measured
Y	1.0	defined benefit obligations
P	1.0	for-sale financial assets
Y	1.0	gross carrying amount
Y	1.0	organized market options
Y	1.0	trading financial assets
Y	1.0	asset backed securities
Y	1.0	master netting agreement
Y	1.0	exempted ccp leg
P	1.0	non-central government public
Y	1.0	european financial stability
Y	1.0	extremely high liquidity
Y	1.0	financial corporate bonds
Y	1.0	deposit guarantee scheme
26Y 3P 1N		

4.5 Abbreviation extraction

We can also briefly look at the abbreviation extraction results in table 9. The results are impressive, indeed for this document the precision is 100%. This shows that extracting abbreviation with our little heuristic is a reasonably straightforward task.

4.6 Performance analysis

Looking at timing performance on figure 7, we tested 20 documents of various lengths with our algorithm and recorded the time taken to extract terms. The test was conducted on a Windows 10 machine running on an Intel i5-4670 3.4GHz CPU supported by 16Go of RAM and a SATA SSD. To our surprise, the time required to perform the term extraction task seems to be linear with respect to document length. More precisely, it seems that it takes about one second to extract terms from a 5,000 words document.

This is relatively quick, at this speed it would take $\sim 200h$ or less than 9 days to analyze the 3.6 Billions words contained in the entire English Wikipedia on our machine. Indeed it took about 14 min to produce this graph 7. For the purposes of DG FISMA, this speed is much more than reasonable since new legislation are not written every day. Such performance is mainly the results of the use of memoïsation in our algorithm.

Table 7: 3-grams extracted using our CPS^2 measure

Custom		
N	0.65	delegated regulation (eu)
Y	0.48	unsecured wholesale deposits
Y	0.35	deposit guarantee scheme
P	0.18	gross holdings included
Y	0.17	multilateral development banks
Y	0.17	legal references and instructions
Y	0.15	own debt securities
Y	0.15	reporting reference date
Y	0.14	extremely high liquidity
Y	0.14	balance sheet exposures
Y	0.13	own covered bonds
Y	0.13	available-for-sale financial assets
P	0.12	for-sale financial assets
P	0.12	non-central government public
Y	0.11	organized market options
Y	0.11	deferred tax liabilities
Y	0.09	european financial stability
Y	0.09	long settlement transactions
Y	0.08	traded debt instruments
Y	0.08	defined benefit obligations
P	0.08	financial liabilities measured
Y	0.08	liquid assets template
Y	0.07	own estimates of lgd
Y	0.07	gross carrying amount
Y	0.07	defined benefit pension
Y	0.07	joint ventures and associates
Y	0.07	financial corporate bonds
Y	0.07	groups of connected clients
Y	0.07	master netting agreement
Y	0.07	central bank repo
25Y 4P 1N		

We tried to do the same experiment by removing memoïsation but after two hours the first document was not analyzed and we gave up. Thus memoïsation speeds up the algorithm by at least a factor of 240 considering the first document was previously analyzed in 30 seconds. We also cannot say exactly how much the algorithmic complexity is affected by memoïsation but clearly the effect is tremendous.

Indeed measure like frequency uses memoïsation to cache the first computation. This first computation needs to count the number of occurrence of a given term in a given entire document. With memoïsation this means a complexity of $O(N + C)$ for C calls of the function computing the frequency. Indeed, only the first call needs to go through the entire document while the next calls only read the cache. Without memoïsation we would have a complexity of $O(CN)$. This is not only true for computing frequency but all the other measures as well which are also cached. Moreover, to compute the C-value of a term we make extensive use of the frequency of other terms, which is where we expect memoïsation to be most effective. It is rational that we were not able to perform the same experiment without memoïsation in a reasonable amount of time.

Table 8: Supergrams extracted

Supergrams	
Y	report net of defaulted exposures
Y	secured lending and capital market
Y	references national gaap compatible ifrs
Y	references national gaap based on bad
N	instructions concerning specific positions
Y	extremely high quality covered bonds
Y	transferable assets representing claims on or guaranteed
N	amount encumbered for a period greater
Y	measured at fair value through profit or loss
N	permitted offsetting short positions in relation
Y	exposure amount of assets that are exposures
Y	leverage ratio exposure value of assets that are exposures
Y	original exposure pre conversion
Y	groups classified as held for sale
Y	financial assets held for trading
Y	financial assets designated at fair value through profit or loss
Y	accumulated changes in fair value
N	group or an institutional protection scheme
Y	financial assets measured at fair value
Y	relevant national gaap based on bad
Y	techniques with substitution effects on the exposure
Y	entail asset encumbrance for that institution
N	loans and advances other than loans on demand
Y	breakdown of financial assets by instrument and by counterparty
Y	deferred tax assets that rely on future profitability
Y	financial liabilities held for trading
Y	losses on financial assets and liabilities designated at fair value through profit or loss
Y	specific allowances for financial assets
Y	regional governments and local authorities
Y	countercyclical capital buffer rate

5 Conclusion

Existing approach for term extraction lack the ability to extract long, or complex terminology. While traditional POS-based methods have a proven record of extracting simple terminology. Our novel unsupervised algorithm proved itself to be state-of-the-art in term of long term extraction. The reason is that we considered long terms to be decomposable in smaller terms which can be extracted using n-grams and concatenated together. To extract n-gram, we have developed a new noise measure (SWF) from which we derived a new term relevancy measure (CPS^2) which reliably and precisely rank terms. Through experiments, we were able to demonstrate state-of-the-art performance with our method for simple and complex terminology extraction. Moreover, we achieve such performance without extreme algorithmic complexity, thus our method is fast. Finally, we have also tested a simple heuristic for abbreviations extraction.

Possible further improvement of this algorithm would be to incorporate a linguistical approach to help the task of term extraction. At the core of this thesis was the desire to prove that a purely statistical method is capable of producing state-of-the-art performance for TE and we believe this was a success. But as we demonstrated, POS-based methods are undoubtedly good at extracting small terms and could be allied with our method for better performance. Linguistic methods can also help extract domain specific stop-word and irrelevant construct. We can also use it as a post-processing filter applied after the n-gram extraction step. For example in a real-life application of this algorithm, we used POS-tagging to remove candidate terms which

Table 9: Abbreviation extracted

Freq	Abbv	Term
40	GS	GROUP SOLVENCY
9	BIA	Basic Indicator Approach
403	IG	Implementation Guidance
148	CRM	CREDIT RISK MITIGATION
263	CB	COMMERCIAL BANKING
4	SFM	Supervisory Formula Method
2	FSB	Financial Stability Board
2623	CR	CREDIT RISK
272	RB	RETAIL BANKING
13	CF	CORPORATE FINANCE
310	CA	CAPITAL ADEQUACY
2595	CR	Credit RiskSee
10	BIA	BASIC INDICATOR APPROACH
2	FEI	Fully Exempt Inflows
425	SA	Standardised Approach
2596	CR	Capital Requirements
6	IAA	INTERNAL ASSESSMENT APPROACH
23	AMA	ADVANCED MEASUREMENT APPROACHES
5	IRBA	Internal Ratings Based Approach
22	AMA	Advanced Measurement Approaches
2	SFA	Supervisory Formula Approach
2602	CR	CAPITAL REQUIREMENTS
23	ASA	Alternative Standardised Approach
4	RBr	RETAIL BROKERAGE
3	ROC	Regulatory Oversight Committee
310	LE	LARGE EXPOSURES
4	DCS	DIRECT CREDIT SUBSTITUTES
45	CTP	Correlation Trading Portfolio
31	IP	IMMOVABLE PROPERTY
106	SME	Small and Medium-sized Enterprises

start or end with a verb. This improvement was not discussed in the thesis as it was too late to incorporate and not in part with the statistical theme.

To conclude, while POS-based methods have proven to be well suited for simple terminology extraction. This thesis demonstrated that statistical method can reach state-of-the-art performance for simple and complex terminology extraction.

Figure 7: Time to extract terms from documents given document length in words

References

[1] David A. Evans and Chengxiang Zhai. Noun-phrase analysis in unrestricted text for information retrieval. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* -. Association for Computational Linguistics, 1996.

[2] Gaël Dias. Multiword unit hybrid extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions analysis, acquisition and treatment* -. Association for Computational Linguistics, 2003.

[3] Fethi Fkih and Mohamed Nazih Omri. Complex terminology extraction model from unstructured web text based linguistic and statistical knowledge. *International Journal of Information Retrieval Research*, 2(3):1–18, July 2012.

[4] Barry Smith, , Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H Scheuermann, Nigam Shah, Patricia L Whetzel, and Suzanna Lewis. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007.

[5] Blaž Fortuna, Nada Lavrac, and Paola Velardi. Advancing topic ontology learning through term extraction. volume 5351, pages 626–635, 04 2008.

[6] Diana Maynard, Yaoyong Li, and Wim Peters. Nlp techniques for term extraction and ontology population. *Ontology learning and population: bridging the gap between text and knowledge*, 167:107–127, 06 2008.

[7] Maria Teresa Pazienza, Marco Pennacchiotti, and Fabio Massimo Zanzotto. Terminology extraction: An analysis of linguistic and statistical approaches. In *Knowledge Mining*, pages 255–279. Springer-Verlag.

[8] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing* -. Association for Computational Linguistics, 1992.

[9] Didier Bourigault and Electlicit De France. Surface grammatical analysis for the extraction of terminological noun phrases, 1992.

[10] Béatrice Daille, Éric Gaussier, and Jean-Marc Langé. Towards automatic extraction of monolingual and bilingual terminology. In *Proceedings of the 15th conference on Computational linguistics* -. Association for Computational Linguistics, 1994.

[11] Steven Bird. NLTK. In *Proceedings of the COLING/ACL on Interactive presentation sessions* -. Association for Computational Linguistics, 2006.

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

[13] Robert M. Fano and David Hawkins. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29(11):793–794, November 1961.

[14] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945.

[16] Denis Nkweteyim. Data structures for information retrieval. pages 1–8, 05 2014.

[17] Andrew MacFarlane. Introduction to modern information retrieval (2nd edition)20045g.g. chowdhury. introduction to modern information retrieval (2nd edition). facet publishing, 2004. 474 pp., ISBN: 1 85604 480 7 london £39.95. *Program*, 38(3):216–217, September 2004.

[18] S. L Zabell. On student's 1908 article "the probable error of a mean". *Journal of the American Statistical Association*, 103(481):1–7, March 2008.

[19] Kenneth Church, William A Gale, Patrick Hanks, and Don Hindle. *Using Statistics in Lexical Analysis*, pages 115–164. 01 1991.

[20] Dongqiang Yang and David Powers. Automatic thesaurus construction. volume 74, 01 2008.

[21] Katerina T. Frantzi and Sophia Ananiadou. Extracting nested collocations. In *Proceedings of the 16th conference on Computational linguistics -*. Association for Computational Linguistics, 1996.

[22] Katerina T. Frantzi, Sophia Ananiadou, and Junichi Tsujii. The c-value/NC-value method of automatic recognition for multi-word terms. In *Research and Advanced Technology for Digital Libraries*, pages 585–604. Springer Berlin Heidelberg, 1998.

[23] Diana Maynard and Sophia Ananiadou. TRUCKS: A model for automatic multi-word term recognition. *Journal of Natural Language Processing*, 8(1):101–125, 2001.

[24] Shahzad Qaiser and Ramsha Ali. Text mining: Use of TF-IDF to examine the relevance of words to documents. *International Journal of Computer Applications*, 181(1):25–29, July 2018.

[25] Adam Kilgarriff and Gregory Grefenstette. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–347, September 2003.