DEGREE PROJECT, IN COMPUTER SCIENCE , SECOND LEVEL
STOCKHOLM, *SWEDEN 2015*

Finding Implicit Citations in Scientific Publications

IMPROVEMENTS TO CITATION CONTEXT DETECTION METHODS

JONATHAN MURRAY

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

KTH Computer Science
and Communication

Finding Implicit Citations in Scientific Publications:
Improvements to citation context detection methods

JONATHAN MURRAY

Master's Thesis at CSC
CSC Supervisor: Olov Engwall
SICS Supervisor: John Ardelius
Examiner: Olle Bälter

September 7, 2015

Abstract

This thesis deals with the task of identifying implicit citations between scientific publications. Apart from being useful knowledge on their own, the citations may be used as input to other problems such as determining an author's sentiment towards a reference, or summarizing a paper based on what others have written about it. We extend two recently proposed methods, a Machine Learning classifier and an iterative Belief Propagation algorithm. Both are implemented and evaluated on a common pre-annotated dataset. Several changes to the algorithms are then presented, incorporating new sentence features, different semantic text similarity measures as well as combining the methods into a single classifier. Our main finding is that the introduction of new sentence features yield significantly improved F-scores for both approaches.

Referat

Detektion av implicita citeringar mellan vetenskapliga publikationer

Detta examensarbete behandlar frågan om att hitta implicita citeringar mellan vetenskapliga publikationer. Förutom att vara intressanta på egen hand kan dessa citeringar användas inom andra problem, såsom att bedöma en författares inställning till en referens eller att sammanfatta en rapport utifrån hur den har blivit citerad av andra. Vi utgår från två nyliga metoder, en maskininlärningsbaserad klassificerare och en iterativ algoritm baserad på en grafmodell. Dessa implementeras och utvärderas på en gemensam förannoterad datamängd. Ett antal förändringar till algoritmerna presenteras i form av nya särdrag hos meningarna (*eng. sentence features*), olika semantiska textlikhetsmått och ett sätt att kombinera de två metoderna. Arbetets huvudsakliga resultat är att de nya meningssärdragen leder till anmärkningsvärt förbättrade F-värden för de båda metoderna.

Contents

Contents

1 Introduction **1**
- 1.1 Problem definition . . . 2
- 1.2 Research question . . . 3
- 1.3 Report outline . . . 4

2 Background and Theory **5**
- 2.1 Citations . . . 5
- 2.2 Text similarity and relatedness . . . 7
- 2.3 Conditional Random Field . . . 12
- 2.4 The classification problem . . . 12
- 2.5 Summary . . . 15

3 Method **16**
- 3.1 Dataset . . . 16
- 3.2 Supervised machine learning approach . . . 18
- 3.3 Graphical approach . . . 25
- 3.4 Similarity measures . . . 30
- 3.5 Combined approach . . . 33

4 Results **34**
- 4.1 Graphical approach . . . 34
- 4.2 Acronyms and Lexical hooks . . . 37
- 4.3 Machine Learning - Cross-validation 1 . . . 37
- 4.4 Information-gain . . . 38
- 4.5 Machine Learning - Cross-validation 2 . . . 39

5 Discussion **41**
- 5.1 Result discussion . . . 41
- 5.2 Conclusion . . . 43
- 5.3 Sources of error . . . 44
- 5.4 Generality . . . 44
- 5.5 Future work . . . 45

5.6 Ethics 45

Appendices **46**

A Word lists **47**
A.1 Athar's determiners 47
A.2 Athar's work nouns 47
A.3 Athar's connectors 47
A.4 Pronouns 48
A.5 Qazvinian and Radev's determiners 48
A.6 Qazvinian and Radev's work nouns 48

B Graphical algorithm results **49**

Bibliography **52**

Chapter 1

Introduction

Citations are an essential part of science. They are used by researchers for many purposes such as giving credit, substantiating their arguments and comparing their work to that of others. However, there is more to a citation than just a pointer between publications. Citations are found in the body text of a paper, and may range from being just a short mention to spanning several sentences. Oftentimes a work is cited in one sentence and then implicitly referred to again in later sentences. Identifying the sentences that relate to a specific reference may be trivial for a human reader but not for a computer. In this project we are specifically interested in **finding all sentences that refer to a specific reference** without using a formal reference notation.[1] These will be referred to as implicit citations (in contrast to explicit citations where a formal notation is used). An alternative view is that the explicit and implicit sentences together make up the *context* of a citation, which helps explain the term *citation context detection*.

There are several applications of solving this problem. First of all, some automated tasks need the citation text as input, and therefore rely on an accurate detection method. For instance, one may want to understand the author's sentiment towards a reference [5], that is whether the author approves of the reference or criticizes it. This is useful since otherwise one may wrongly assume that citations are always positive indicators. The citation text may also be used as source material for creating a summary of the reference [29]. In many cases, reading what others say about a publication may give a more true picture of its qualities than reading the author's own summary. These tasks both seem to benefit from including not only explicit citation sentences but also implicit ones. Another possibility is to present the citation sentences directly to a user. Search engines such as Google Scholar [11] and AAN (ACL Anthology Network) [19] let users search for publications and also list incoming citations from other publications. On AAN the explicit citation sentences are also shown. This feature could be improved further by including additional related sentences to form the complete story of what the author

[1] Typically, explicit citations follow some standard notation style. For instance, using the Harvard style, the authors' family names and the publication date are listed inside a parenthesis.

says about the reference. Athar and Teufel found that especially negative opinions about others' work seldom appears in explicit citations [5]. By including implicit citations, the amount of negative sentiment that they found was increased by more than 300%.

This thesis extends upon two state-of-the-art methods to solve the problem of citation context detection. The methods have not been evaluated on the same dataset, so it is not clear beforehand if one performs better than the other. They both come from the highly regarded *Association for Computational Linguistics*. In the first one, Qazvinian and Radev [29] use a Conditional Random Field to model the sentences and an iterative algorithm to find the probability that each sentence is an implicit citation. In the second one, Athar and Teufel [5] use handcrafted sentence features and machine learning to classify the sentences. In this thesis, the two methods are extended with new features and semantic similarity measures, and are also combined with each other. The performance of the proposed methods are evaluated using the annotated dataset [3] from Athar and Teufel's work [5]. The dataset consists of papers that have cited a selection of 20 distinct papers, with all sentences annotated as either (1) *explicit citation*, (2) *implicit citation* or (3) *not citation*.

1.1 Problem definition

Formally, we are dealing with a classification problem. Each sentence of a research paper can be classified, in relation to a given referenced paper, as one of the following (1) *not a citation* (2) *implicit citation* and (3) *explicit citation*. See Figure 1.1 for an illustration. Finding sentences of class (3) is trivial, and therefore we only consider the other two classes as output of the algorithms. Figure 1.2 gives an overview of the input and output for the problem. Since a sentence that cites a paper is likely to have some similarities with the paper itself, the full-text of the paper will be useful in the method.

grams. In order to improve sentence-level evaluation performance, several metrics have been proposed, including ROUGE-W, ROUGE-S (Lin and Och, 2004) and METEOR (Banerjee and Lavie, 2005). ROUGE-W differs from BLEU and NIST

[...]

METEOR is essentially a unigram based metric, which prefers the monotonic word alignment between MT output and the references by penalizing crossing word alignments. There are two problems with METEOR. First, it doesn't consider gaps in the aligned words, which is an important feature for ...

Figure 1.1: [5] Sentences coloured according to their citation context classes, with respect to Banerjee and Lavie. (1) not citation , (2) implicit citation and (3) explicit citation .

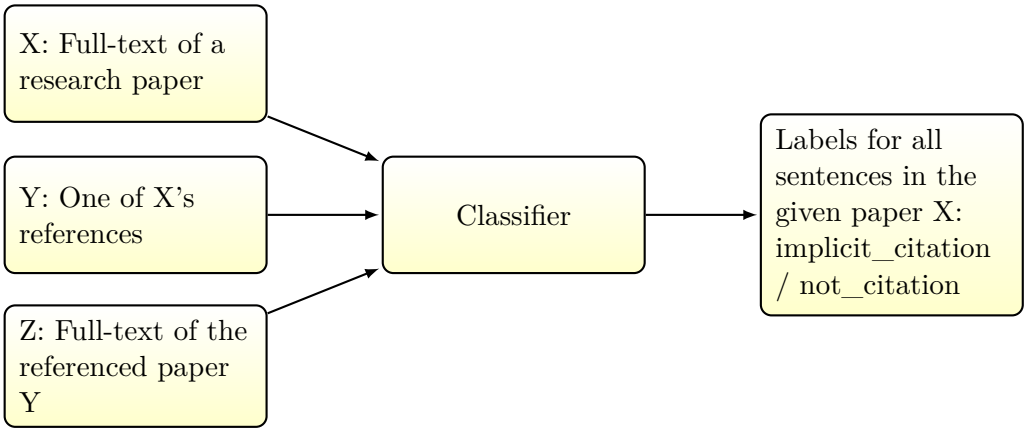


Figure 1.2: Input/output for the problem.

1.2 Research question

The problem outlined above will be solved using two distinct approaches, namely a supervised machine learning classifier and an iterative algorithm based on a graphical model. The research question is if these existing methods can be improved, either by adding more features or through more sophisticated text similarity measures, and additionally if the performance can be further improved by combining the methods. The *performance* is evaluated with F_1- and F_3-values. It is an explorative and somewhat open-ended question.

1.3 Report outline

The structure of this report is as follows. Chapter 2 serves as a background to the subject, mainly explaining citation context and different Natural Language Processing techniques. In chapter 3 the dataset and the different approaches are presented. Finally, chapter 4 presents the experiment results and chapter 5 concludes the thesis with a discussion and proposed future work.

Chapter 2

Background and Theory

2.1 Citations

Citations are used by researchers for many purposes, but the most common is to show that the reference influenced the author's work. Similarly to how PageRank [27] has been used to measure the influence of websites based on their links, different metrics based on citations have been used to measure the influence of research papers.

However, many flaws have been pointed out about using citations alone as a measure for influence. Moravcsik and Murugesan [25] found that 40% of citations in scientific publications are perfunctory, meaning they are only general acknowledgements. Furthermore, Simkin and Roychowdhury [34] tried to estimate how many researchers actually read the papers they cite, based on reoccurring misprints in citations. They arrived at the alarmingly low estimate that researchers only read 20% of the cited papers. Taking this into account, it is questionable if a citation alone can be seen as a sign of influence.

There has been much research about the characteristics of citations and several subfields have arisen. On the one hand, there is citation analysis which is concerned with the quantitative analysis of citations. This includes techniques such as co-citation which measures the similarity of papers by how often they are cited together, and h-index [17] which rewards researchers for having many highly cited papers[1]. On the other hand, there are techniques that are concerned with the qualities of citations within a single paper, rather than a quantitative analysis of inter-paper relations. Zhu et al. [38] showed that simply counting the number of times a paper is cited is a strong predictor of the influence level of the reference. Others look at individual citations and analyse the related text, which in the literature is referred to as the citation context.

[1]The formal definition is: "A scientist has index h if h of his/her N_p papers have at least h citations each, and the other $(N_p - h)$ papers have no more than h citations each".

2.1.1 Using citation context

The ability to identify citation contexts has many applications. Teufel et al. [35] match manually constructed cue-phrases against the citation context along with other features to classify the author's intent of a citation. Four categories of intent are defined: (1) pointing out a weakness about the reference, (2) comparing one's own work to the reference, (3) using the reference as a basis for one's work and (4) objectively describing the reference. Athar and Teufel [5] use several language features of the citation context for sentiment analysis. Having this sentiment information can give a more nuanced and true picture of how a research field is structured, as positive citations indicate influence in a way that negative citations clearly do not. Qazvinian and Radev [29] apply citation context detection to the problem of creating summaries of research topics, commonly known as surveys. The general idea is that a paper can be summarized by what others have said about it rather than by the contents of the paper itself. They show that using the entire citation contexts – compared to using only the explicit citation sentences – yields improved survey generation. Ritchie [31] shows that, in information retrieval, a document representation that incorporates citation contexts can increase the retrieval effectiveness. The works mentioned above are all dependent on a way to extract the citation context.

2.1.2 Defining citation context

How does one define the context of a citation? A very simple approach, used with some variations by Ritchie [31] and Mei and Zhai [24], is to start at the explicit citation and extend the context a predefined distance in both directions, possibly truncated at sentence delimiters and other citations. However this method does not consider topic shifts and has no way of correctly setting the context boundary other than choosing a boundary that is statistically likely. Furthermore, it ignores situations where the context is split up by unrelated sentences. Qazvinian and Radev [29] argue that although the majority of context sentences have no gaps between them, there are counterexamples (in their dataset 22 out of 295), which should not be neglected.

If one decides to consider discontinuous contexts, there is still the question of how fine-grained the detection should be. Abu-Jbara and Radev [2] focus on sentences that contain several different citations, and use NLP-techniques to extract fragments of the sentence that belong to the target citation. In contrast, in the work of Athar and Teufel [5] only whole sentences are considered, and one sentence may be part of several contexts. Figure 2.1 shows an example of a discontinuous citation context.

grams. In order to improve sentence-level evaluation performance, several metrics have been proposed, including ROUGE-W, ROUGE-S (Lin and Och, 2004) and METEOR (Banerjee and Lavie, 2005). ROUGE-W differs from BLEU and NIST

[...]

METEOR is essentially a unigram based metric, which prefers the monotonic word alignment between MT output and the references by penalizing crossing word alignments. There are two problems with METEOR. First, it doesn't consider gaps in the aligned words, which is an important feature for ...

Figure 2.1: Implicit citations example from Athar and Teufel [5]. The first sentence is an explicit citation to Banerjee and Lavie. Later in the text three consecutive sentences again refer to their work, without explicitly citing the authors. We call these sentences implicit citations.

2.1.3 Detecting implicit citation sentences

In this work we limit ourselves to identifying whole sentences that belong to the context of a specific citation. Many methods have been proposed to solve this problem. Qazvinian and Radev [29] use a Conditional Random Field and an iterative algorithm. Another approach, used by Athar and Teufel [5], is to find the sentence labels by training a Machine Learning classifier on a set of hand-constructed features. We will extend upon these methods in Chapter 3.

2.2 Text similarity and relatedness

As we are working with text in general, and specifically trying to model characteristics of sentences, text similarity and relatedness has a role to play. In prior work this has been limited to n-grams but this thesis goes further and extends the method with a number of alternative measures. An overview of different techniques is given below, after a short discussion of terminology.

Finding the similarity and relatedness of texts is an important and well-established problem within the fields of natural language processing and information retrieval. However, it is worth noting that similarity and relatedness are distinct concepts, as Resnik [30] points out. Semantic similarity often arises from two words being instances of the same class, such as bikes and cars both being vehicles, whereas there are many other word relations that lead to semantic relatedness, such as meronymy (car-wheel) and antonymy (hot-cold). In fact Morris [26] found in an experimental study that words do not even have to be part of these relations to seem related to a human. Rather, the majority of the perceived relations were ad-hoc relations such as ”words related to funerals” or ”positive human characteristics”.

2.2.1 Simple word-based methods

Below, some simple text similarity measures used in this project are explained briefly.

n-grams

n-grams is a technique with many applications within Natural Language Processing. The idea is to measure not only occurrences of single words but also sequences of words. For example a bigram ($n = 2$) or trigram ($n = 3$) denotes two or three contiguous words in a sequence. In general, n-grams are not limited to words but can model any items occurring in a sequence, for instance single characters. For text similarity n-grams are used to measure which words and phrases occur in both texts. Unfortunately, an inherent problem of n-grams is data sparsity. Increasing n drastically reduces the probability of observing specific n-grams in a corpus. Brown et al. [7] found that even when using a corpus of more than 366 million words, 14.7% of all trigrams in a new text sample can be expected to be absent from this corpus.

skip-grams

An example from Guthrie et al. [13] illustrates one aspect of the sparsity problem. Consider the sentence "I hit the tennis ball.". The three trigrams extracted are "I hit the", "hit the tennis" and "the tennis ball". Now consider another similar sentence "Hit the ball!". If we consider n-grams with $n \geq 2$ the only common n-gram is "hit the". However, one could argue that the trigram "hit the ball" is in fact present in both sentences, although it is split up in the first one.

To handle this kind of sparsity one can allow "gaps" in the n-grams. A k-skip-n-gram is defined as an n-gram with up to k words (including 0) skipped between the other words. For example extracting 1-skip-trigrams from the first sentence above would result in "I hit tennis", "I the tennis", "hit the ball", "hit tennis ball" in addition to the original trigrams. See Figure 2.2 for an illustration. Allowing for even more gaps increases the number of extracted skip-grams drastically. For instance, a sentence of 20 words contains 18 regular trigrams, 53 1-skip-trigrams and 100 2-skip-trigrams [13].

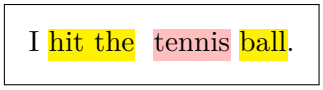


Figure 2.2: Skip-1-trigram "hit the ball" extracted from the sentence "I hit the tennis ball.". The word "tennis" is skipped, which would not be allowed using regular n-grams.

tf-idf

A well-known statistic for text similarity is tf-idf [33], short for term frequency-document frequency, which models documents by term frequencies, giving extra weight to rare terms that do not appear in many different documents. Expressed simply, for each document it is noted how many times each unique word appears. Additionally, for each word it is noted how many documents it appears in. Finally, for each document-word pair a score is computed from these counts. There are several accepted ways of computing this score. The formula that will be used in this work is

$$tfidf = (1 + log f_{t,d}) * log(1 + \frac{N}{n_t})$$

where $f_{t,d}$ is the number of times the word appears in the document and $\frac{N}{n_t}$ is the fraction of documents that contains the word. When comparing documents with this representation, having high scores for the same words indicates similarity. This has proved to be a good similarity measure based on exact matches, but it fails to detect the usage of synonyms or related words. For this, a more sophisticated approach may be needed.

2.2.2 Distributed word-vector models

Several suggestions have been made to overcome the limitations of tf-idf. One approach is to go beyond the definition of a word as simply a unique string of characters, and instead give them a representation that is based on the contexts they tend to appear in. Specifically, this is achieved by processing a large text corpus and, for each unique word, count other words that occur in its proximity. Thus each word is represented by a vector that describes these qualities. These methods build upon the assumption that the semantic similarity of two words is correlated to the similarity of the contexts they tend to appear in [32]. The assumption is intuitively pleasing as, for instance, *motorbike* and *bicycle* are two very related words that may tend to appear in similar contexts, such as ”I parked my motorbike/bicycle outdoors” or ”riding a motorbike/bicycle”.

Latent Dirichlet Allocation

One notable example of context based similarity measures is Latent Dirichlet Allocation (LDA), proposed by Blei et al. [6] as a probabilistic way to model documents[2] as distributions over topics[3]. The topics are, in turn, distributions over words, so that a document's probability for a given topic measures the likelihood that the document's words were generated by the topic. A variational EM-algorithm (Expectation Maximization) is used to find the parameters that maximizes the likelihood

[2]Document is an abstraction that can mean a sentence, or any collection of words, in addition to an actual text document.

[3]The words *concept* and *topic* have been used interchangeably in the literature.

of the data. Later this method was improved by Blei and Lafferty [21] to take topic correlations into account, which led to better topic fits and a capacity for a higher number of topics.

Latent Semantic Analysis

Latent Semantic Analysis (LSA) is similar to LDA in that words are modelled by their contexts, but in this case a dimensionality reduction is performed using Singular Value Decomposition (SVD). First a matrix is built with rows representing single words, columns representing contexts and the cells denoting occurrence counts. Next, this matrix is factorized to three matrices, one describing the rows, one describing the columns and one containing scaling values. Although it is possible to construct three matrices that when multiplied gives back the original matrix, typically this preciseness is given up for the sake of dimensionality reduction, and instead a least-square fit of the original matrix is obtained upon multiplication. The dimensionality tends to be high, up towards a few thousand. [22]

2.2.3 Explicit topics

One alternative approach is to use explicit topics that are predefined by humans. A clear advantage of this is readable easy-to-understand topic names. Some kind of database with predefined topics is then needed. Of the two options presented below, only Wikipedia is used in this work.

Wikipedia as a knowledge source

Gabrilovich and Markovitch [10] propose a new method "Explicit Semantic Analysis" that maps input texts to high dimensional vectors with entries corresponding to Wikipedia articles. Intuitively the vector weights denote how strongly related the input text is to all articles, based on the overlap of notable words, i.e. words with a high tf-idf score. For semantic text relatedness they show an improvement over earlier methods.

More attempts have been made to use Wikipedia as a source of topics. Milne and Witten [37] propose a different method where relatedness is defined by the hyperlinks between Wikipedia articles rather than the article contents. Although the performance does not beat that of Gabrilovich and Markovitch [10], they point out that this method requires far less data and resources, since there is no need for the full-text of the articles. An open source implementation has been released under the name wikipediaminer [1].

An additional advantage of using Wikipedia as a knowledge source is that there is no need to construct topic relations manually, as this is done by the community by linking articles.

WordNet as a knowledge source

Another famous database that has been used for semantic text comparison is WordNet [36], which provides word definitions, word senses, and relations between senses. A word sense, in WordNet called a *synset*, is a semantic entity that is usually shared by multiple words. For instance, *car* and *automobile* both describe the same entity. Of course, most words also have several different meanings and therefore correspond to several synsets. The relations between the synsets can be visualized as a graph, with a small number of abstract concepts such as *artifact* or *entity* high up in the graph, and specific nouns such as *notebook* on lower levels. Budanitsky and Hirst [8] explored several different measures of semantic similarity using this graph. They found that a measure by Jiang and Conrath [18] consistently outperformed the others, which was later supported by four other papers. The proposed measure combines an edge-based approach, where the similarity of two synsets is based on their distance in the WordNet-graph, with an information content approach, where the similarity is based on how much information is shared by the synsets.

There are some difficulties about applying these WordNet-based measures to longer texts. First of all, it is common with quite a few rare synsets listed for every word. As an example, below are some of the synsets for the word *dog*, taken from WordNet.

- (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) "the dog barked all night"
- (a dull unattractive unpleasant girl or woman) "she got a reputation as a frump"; "she's a real dog"
- (informal term for a man) "you lucky dog"
- (someone who is morally reprehensible) "you dirty dog"

There is therefore a problem of going from a sentence of words to a sentence of synsets. One approach is to simply take the most common synset for every word; there is some limited information about synset rarity in WordNet. However, in some cases the most common synset is not the one intended by the author, and for some words there may be several common synsets.

There is also the issue of computational cost. If one synset is extracted for every word, and two long texts are to be compared to each other, how should this be done? One approach would be to simply look for identical synsets, but this would only help to find synonyms, and not related words. If one wants to use a more advanced measure such as graph distance, which synsets should be compared? If the answer is "all of them", and the sentences consist of n words, we are dealing with n^2 graph distance computations, for every single sentence comparison. A cheaper way would be to only compare some of the synsets for each sentence, but it is not clear how these would be chosen.

For the reasons presented above, WordNet will not be used as a knowledge source in this project.

2.3 Conditional Random Field

Conditional Random Field (CRF) is a graphical model that serves as a theoretical basis for the graphical method described later in this report. Similarly to a Hidden Markov Model, its task is to classify a sequence of items given their observed values. There are two disjoint sets of nodes, representing the observations and corresponding hidden labels. Since the point of the model is to predict the hidden nodes given the observed nodes, one can assume independence between the observed nodes without any downside. For the hidden nodes a Markov assumption is made, that they only depend on a limited set of neighbours. In figure 2.3, these dependencies are represented by the horizontal edges. Formally they are known as compatibility functions, whereas the values of the observed nodes are called potential functions. In the context of this work each sentence corresponds to a pair of nodes, and the label to be predicted is whether the sentence is an implicit citation or not.

Finding the true labels of the data implies finding a state of maximum stability, given the compatibility functions and potential functions. One possibility is to use iterative methods until convergence is reached.

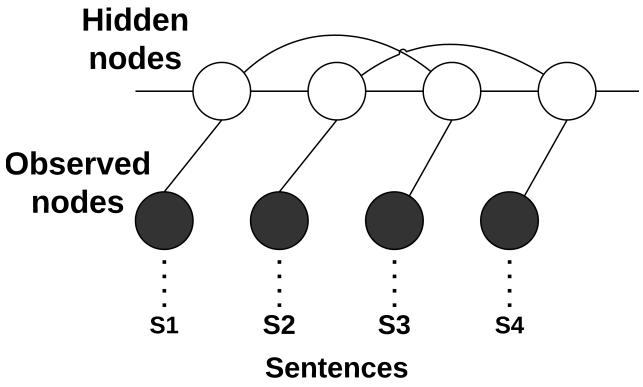


Figure 2.3: Conditional Random Field example

2.4 The classification problem

Machine Learning is a form of Artificial Intelligence where a computer solves a non-trivial problem without being instructed by a programmer as to exactly how the problem should be solved. In this project we are specifically interested in the

problem of classification, as we aim to classify sentences as either *implicit citations* or *not citations*. The machine learning method used in this thesis deals with a certain form of classification known as *supervised classification*. The analogy is that the program practises while being supervised by a human teacher that gives continuous feedback. In reality, the data has been preannotated by a human and the "feedback" is extracted automatically by an algorithm. In this work, the classifier learns from a dataset of sentences that have been annotated as *not citation*, *implicit citation* or *explicit citation*.

2.4.1 Classification evaluation

Now we consider how one evaluates the performance of a classifier. Naturally, making correct classifications should mean a higher performance and vice versa. Below, some important terminology is introduced and a common evaluation metric is presented. We are limiting ourselves to only two classes, although in some problem domains one may want three or more.

positive/negative The classes. Usually one class is more interesting than the other (in our case implicit citations) and that one is called the positive class.

true positive A positive instance that is correctly classified as positive.

true negative A negative instance that is correctly classified as negative.

false positive A negative instance that is wrongly classified as positive.

false negative A positive instance that is wrongly classified as negative.

tp/tn/fp/fn These variables are used after a classification to describe how the classification went. They denote the number of instances classified according to the patterns presented above. For example, tp denotes the number of true positives in the classification.

confusion matrix A 2x2 matrix that displays tp, tn, fp and fn of a classification. See Figure 2.4 for an example.

precision Measures the ratio of correct classifications among the instances that were classified as positive.

$$precision = \frac{tp}{tp + fp}$$

recall Measures the ratio of positive instances that the classifier managed to find.

$$recall = \frac{tp}{tp + fn}$$

$\mathbf{F_1}$ A measure that combines precision and recall. To get a good F_1-value a classifier must strike a balance between the two.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

$\mathbf{F_\beta}$ A generalised version of F_1 that puts more emphasis on recall than precision (if $\beta > 1$).

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{\beta^2 * precision + recall}$$

k-fold cross validation Cross validation is a technique for evaluating the performance of a classifier using a set of training data. First, the data is split up into k folds. Then, for each fold, the classifier is trained on the other $k - 1$ folds and evaluated on the left-out fold. The average performance of the k evaluations is a statistically reliable measure if k is big enough.

tp	fn
fp	tn

(a) The structure of the confusion matrix.

10	5
2	4

(b) There are 15 positive instances. 10 of them are correctly classified as positive and 5 are classified as negative. Of the 6 negative instances, 2 are wrongly classified as positive.

Figure 2.4: A small confusion matrix example

F_1 is a common metric for measuring the performance of a classifier. In this work, F_3 is also used, mostly to compare the results to prior work that only presented F_3-values. Aiming for a high F_β ($\beta > 1$) can especially be valid when it is important to find as many positive instances as possible, even if that means more incorrect classifications. Examples where this may be true includes disease detection or malfunction detection in aeroplanes, where a false negative can spell disaster.

2.4.2 Combining several classifiers

Combining several different classifiers into one is a well-known method for increasing accuracy. The classifiers are often referred to as an *ensemble of classifiers*, and a common method is to simply let the classifiers vote on the labels of unseen instances. [9]

Hansen and Salamon [14] (1990) explored the use of ensembles for neural networks. They found that using an ensemble of neural networks with plurality voting - where the classification proposed by the majority of the classifiers is chosen - can be expected to perform far better than a single network. In the case of majority voting - where the chosen classification must have been proposed by more than half

of the classifiers - they showed that, if the error rate of each classifier is $p < 1/2$ and the classifiers' responses are independent, then the more classifiers we add, the higher accuracy we will get. Of course, in our work we are dealing with a binary classification problem which makes majority voting synonymous to plurality voting.

2.5 Summary

This chapter has introduced prior work that is central to the project. Citations have been discussed and specifically citation context and its different definitions, applications and extraction methods. Some general Natural Language Processing techniques and classification terminology has also been explained, as well as a graphical model that is built upon in later chapters.

Wikipedia and WordNet were presented as two possible knowledge sources to build a semantic similarity measure upon. A choice was made to only consider Wikipedia in this project, due to the difficulties of applying synset similarity to text similarity of long texts.

Chapter 3

Method

The problem of finding implicit citations is solved using two distinct approaches, namely a supervised machine learning method extending the work of Awais and Athar [5] and a graphical model extending the work of Qazvinian and Radev [29]. Additional features as well as semantic similarity measures are added to the prior works. The two approaches are also combined in the hope that this will yield an even better result. This chapter starts out with an overview of the dataset, followed by descriptions of the mentioned methods.

3.1 Dataset

All experiments in this work are run on the dataset [3] collected and annotated by Athar [5]. The dataset was provided as a set of HTML-files that show the distribution of citations visually as a coloured grid. Thus, a parser was written in Java to extract only the information that was relevant for this project. An analysis and discussion of the dataset is given below.

The dataset consists of the full text of 852 papers which cite 20 selected papers. All papers are from the ACL Anthology Network (AAN) corpus. In total there are 203,803 sentences and 1,034 paper-reference pairs. All sentences have been marked as (1) *not a citation* (2) *implicit citation* or (3) *explicit citation.* Of all the sentences, less than 1% belong to the positive class (implicit citations), so the dataset is extremely skewed. For the citation sentences the sentiment towards the reference (negative, positive, objective/neutral) has also been annotated by Athar, but this is not used in this project. Figure 3.1 shows a tiny part of the dataset structure, also highlighting the many-to-many relation of citations.

The distribution of citation sentences was studied for the dataset. Specifically the gaps between implicit citations and other citations are presented in table 3.1. Similarly to Qazvinian and Radev's dataset [29] the vast majority of implicit citations are directly adjacent to other citations (implicit or explicit).

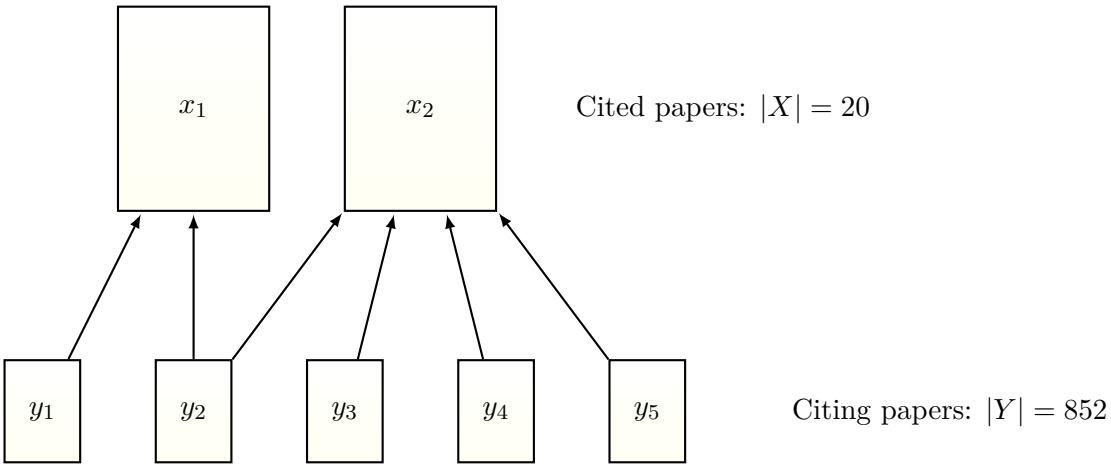


Figure 3.1: Overview of the dataset. $X = x_1, x_2, ..., x_{20}$ denotes the set of cited papers, and $Y = y_1, y_2, ..., y_{852}$ denotes the papers citing these. An arrow from y_i to x_j means that paper y_i cites paper x_j.

Table 3.1: Distribution of gaps between implicit citations

gap size	0	1	2	3	4	5	6	7	8	9	> 9
instances	1164	209	90	62	46	19	17	21	9	11	111

3.1.1 Dataset discussion

There are a few obvious problems with the dataset. First of all the separation of sentences is far from perfect. Period characters seem to be used as delimiters and in many cases one sentence that contains many periods is wrongly split up. In other cases several sentences get merged into a single sentence due to a lack of punctuation. As an example of the latter, figure 3.2 shows the start of paper J93-1002. In the dataset this text was given as a single sentence, even though it is actually a title, two author names and the beginning of an abstract.

Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars

Ted Briscoe* John Carroll*

We describe work toward the construction of a very wide-coverage probabilistic parsing system for natural language (NL), based on LR parsing techniques. [...]

Figure 3.2: The beginning of paper J93-1002

675 References Steven Abney.
2004.
Understanding the Yarowsky algorithm.
CL, 30(3):365395.
Christoph Arndt.
2001.
Information Measures.
Springer.
John Blitzer, Ryan McDonald, and Fernando Pereira.
2006.
Domain adaptation with structural correspondence learning.
In EMNLP, pages 120128.

Figure 3.3: The first 3 references of D07-1070. Each line is a sentence in the dataset.

Another problem is that in many cases the reference section at the end of a paper is included in the dataset. This does not suit our problem definition of classifying the sentences of a paper. The ”sentences” contained in the reference section are not interesting to us. What's worse is that the frequent use of punctuation in references results in a high number of sentences. As an example, figure 3.3 shows the beginning of the reference list for paper D07-1070. Here, every reference results in four irrelevant sentences, corresponding to the author name(s), year, title and journal. In total, the reference list gives rise to more than 170 sentences (out of roughly 450 for the whole paper) that are all marked as *not citation*. This is especially problematic, since intuitively a ”sentence” that consists of only the title of another paper would probably be considered a citation to the paper, but here the classifier is being told the opposite.

To try to prevent this problem, a Java-method was written to detect the start of a reference section, using regular expressions. When parsing the dataset, whenever a probable reference header is detected the rest of the paper is ignored.

3.2 Supervised machine learning approach

In this section the machine learning approach is presented, which extends upon the work of Athar and Teufel [5]. Sentences are classified according to two classes (1) *not a citation* and (2) *implicit citation*. Explicit citation sentences are known beforehand and are not given to the classifier. Below, a short description of Athar and Teufel's work is given together with their presented results, followed by the contributions of this thesis.

3.2.1 Athar and Teufel's work

Athar and Teufel used a number of hand crafted features to model the sentences. They are shown in Table 3.2. Athar also evaluated the information gain of individual features and found that the three features with the highest information gain by far was (in this order) `LEXICAL_HOOK(i)`, `ACRONYM(i)` and `CITE(i-1)` [4].

Table 3.2: Sentence features used by Athar and Teufel [5].

Feature name	**Athar and Teufel's definition**	**Interpretation in this thesis**
CITE(i-1)	S_{i-1} contains the last name of the primary author, followed by the year of publication within a four-word window.	The fact that the previous sentence says a lot about the current sentence, as most implicit citations occur right after an explicit citation.
CITE(i)	S_i contains the last name of the primary author followed by the year of publication within a four-word window.	Not used in this work. The explicit sentences are known from the start and not captured using features.
AUTHOR(i)	S_i contains the last name of the primary author.	Captures cases where the cited author is referred to, but not explicitly.
OTHER_CITE(i)	S_i contains a citation other than the one under review.	May be a negative indicator although it is of course possible to cite several papers in one sentence.
ACRONYM(i)	S_i contains an acronym used in an explicit citation.	Acronyms such as HMM (Hidden Markov Model) or IR (Information Retrieval) are not uncommon when discussing others' work in research papers. This feature registers acronyms that are commonly used in the context of the cited paper, and then looks for occurrences in the sentence that is being classified.
LEXICAL_HOOK(i)	S_i contains a lexical hook.	Lexical hooks are defined as a phrase with one or more capitalized words such as "Hidden Markov Model" or "Xerox". Similarly to ACRONYM(i) the purpose of this feature is to consider common ways of referring to a source. Sometimes the most common lexical hook directly corresponds to the most common acronym, but to handle other cases the acronym variant is also constructed and stored for each lexical hook, even if it has not been observed.
DET_WORK(i)	S_i contains a determiner followed by a work noun.	"Work nouns" are typical words that describe some work or research. They are combined with determiners to create bigrams such as "this work", "such algorithms" or "that method". See Appendix A for word lists.
PRONOUN(i)	S_i starts with a third person pronoun.	May indicate that this sentence continues the subject of the previous sentence. See Appendix A for word lists.
CONNECTOR(i)	S_i starts with a connector.	Connectors are words that bind the sentence semantically to a previous sentence. Examples include "however", "although" and "thus". See Appendix A for word lists.
HEADING(i-1)	S_{i-1} starts with a (sub)section heading.	An author may be less likely to use an implicit citation in the beginning of a (sub)section.
HEADING(i)	S_i starts with a (sub)section heading.	
HEADING(i+1)	S_{i+1} starts with a (sub)section heading.	
NGRAMS	"We also add n-grams of length 1 to 3 to this lexical feature set [...]"	n-grams are extracted automatically using Weka.

Athar and Teufel's results

To evaluate the classifier, Athar and Teufel used 10-fold cross validation. All the data was split up into 10 folds using Weka *stratification* which ensures that the folds have the same class distribution as the whole dataset. A Support Vector Machine was used as a classifier. To ensure that a high score is not gained by classifying the trivial explicit citations, all *explicit citation*-labels were changed to *not citation*. Their results are presented in Table 3.3 and Table 3.4. For the purpose of easy comparison to other approaches, in this thesis the F_3-score of their results was also computed and added.

Table 3.3: Athar and Teufel's **n-gram baseline**

	implicit citation	**not citation**
Precision	0.299	0.997
Recall	0.447	0.994
F-score	0.358	0.995
F_3-score	0.426	0.994

Table 3.4: Athar and Teufel's **sentence-features**

	implicit citation	**not citation**
Precision	0.497	0.996
Recall	0.529	0.996
F-score	0.513	0.996
F_3-score	0.526	0.996

It is clear from the presented results that the features improved the performance. However a brief discussion about overfitting seems appropriate. There is a potential problem with the way that the folds were extracted for cross-validation. Although the nature of cross-validation ensures that the same instance can not occur both in training and test data, it is still possible for correlated instances to do so. Every instance (sentence) comes from a specific paper. Letting the classifier train on other sentences from the same paper, which will likely be the case using this method, may be an unfair advantage and not reflect a typical real-world scenario of using the classifier. In the case of the n-gram baseline the classifier may learn phrases and expressions that this specific author uses to refer to the target paper. The same may be true for the `ACRONYM` and `LEXICAL_HOOK` features. An alternative evaluation method is presented in section 3.2.2.

3.2.2 Contributions of this thesis

This work introduces a new set of features in addition to the ones proposed by Athar and Teufel. The features were thought up by reasoning about what qualities

an implicit citation sentence displays that separates it from other sentences, and inspiration was taken from the existing features. Feature selection is often an important part of machine learning algorithms, and these features are added with the hope of improving the performance of the classifier.

The new features are presented below with short explanations. Some of them are simple extensions of existing features, such as `CITE(i-x)` and `CITE(i+x)` which are generalized versions of `CITE(i-1)`. The new similarity-based features are intended as a more sophisticated way to handle the lexical aspects of a sentence compared to n-grams that were used in prior work. The last two features, which look for determiners in the sentence, are meant to make up for the strictness of `DET_WORK(i)` which is limited to a predefined list of *work nouns*[1].

CITE(i-x) measures the distance to the closest previous explicit citation. This is a generalized version of `CITE(i-1)` in that it checks if the previous sentence is an explicit citation, but also considers sentences further back in the text. A max value of 4 was chosen, so if no explicit citation is found among the 3 previous sentences, the feature is set to 4. What makes `CITE(i-1)` a useful feature is that the majority of implicit citations occur right after an explicit citation. However, the tendency of implicit citations to co-occur with explicit ones stretches beyond one sentence, as seen in Figure 3.1. Looking for explicit citations several steps backwards in the text should help find implicit citations that are not targeted by `CITE(i-1)`.

CITE(i+x) measures the distance to the closest following explicit citation. There is no similar feature in the original set. Similarly to `CITE(i-x)` a max value of 4 is used.

TITLE_SIMILARITY(i) denotes the similarity of this sentence to the title of the cited paper. The reasoning is that the title may include terms that are quoted directly when citing the paper.

CONTENT_SIMILARITY(i) denotes the similarity of this sentence to the content of the cited paper.

CITE_SIMILARITY(i) denotes the similarity of this sentence to other sentences in the dataset that are explicit citations to the target paper. It seems intuitive that sentences that cite the same paper should have some similarities, and since the explicit citation sentences are given (or trivial to find in a real-world scenario) they can be used for this comparison. To implement the comparison a union of all explicit citations is constructed.

STARTS_DET(i) is a binary feature that denotes whether the sentence starts with one of the determiners given in Appendix A. This is a relaxed version of `DET_WORK(i)` that may catch unpredicted expressions.

[1] See Appendix A.2

`CONTAINS_DET(i)` is an even more relaxed version of `STARTS_DET` where the determiner can occur anywhere in the sentence.

For all the similarity-based features, different similarity measures can be used, such as tf-idf cosine-similarity and more advanced semantic vector approaches.

Implementation of features

Most of the features, both the original and the new, are trivial to implement with simple word comparisons using predefined word lists to match *determiners* and *work nouns* and regular expressions for matching headings. Two features that require some attention however, are `ACRONYM(i)` and `LEXICAL_HOOK(i)`. Since these were also pointed out by Athar as having the highest information gain [4], the choice of acronyms and lexical hooks can be assumed to have a big impact on classifier performance. There are some important choices to be made when extracting acronyms and lexical hooks, which will be discussed below.

The point of both features is to register common ways of referring to the target paper by analysing explicit citations to the paper. We chose a window size of 80 characters in each direction when extracting phrases close to an explicit citation. However, one must consider the risk of false positives and negatives. There are several ways that an irrelevant expression can be chosen if special care is not taken to prevent this. One source of trouble is author names. Consider the example of finding the phrase *Xerox* that was often used when referring to a specific paper. If the paper is mentioned together with other papers and other authors, how does the algorithm know which capitalized words are in fact author names and should not be considered lexical hooks, and reversely how does it know that *Xerox* is not an author name? A function was written to predict if a given word in the sentence is an author name or not. The following details were considered:

- Are all characters capitals (ignoring any trailing s)? The word is most probably an acronym rather than a name.
- Is the next word ”et”? This is most probably an author name in a context like ”Johnson et. al (1995)”.
- Does the next word appear to be a year? This may have a higher chance of being an author name.
- Is the next word ”and”? The word may be an author name in a context like ”Johnson and Wilson (1997)”.

Since the function presented above is far from perfect and only makes a best-effort prediction, it is not adequate to discard a word immediately based on the prediction. Especially the last feature runs a high risk of excluding relevant matches. It is easy to imagine a case such as ”Johnson (1993) used both a Support Vector Machine and other classifiers.” where Support Vector Machine would be predicted

as an author name. This weakness is handled by registering the number of positive and negative predictions for each word, and then discarding words with a majority of negative predictions.

The next step is to pick out the most relevant of all found matches. This is done using a similar statistical approach, explained below.

1. First all matches with less than three occurrences are removed.
2. Then the matches are sorted based on three occurrence measures:
   - E: Number of occurrences in explicit citations
   - $\hat{E}$: Number of occurrences in other sentences
   - C: Number of occurrences in the cited paper
3. The score used for sorting is defined as $S = \frac{E * C}{\hat{E}}$

Last of all, the top N matches are chosen. For this work a value of $N = 2$ was chosen. In Athar and Teufel's work [5] `ACRONYM(i)` and `LEXICAL_HOOK(i)` were binary features. Here however, a max score of 1 is given if the most relevant expression is present and $1/n$ if the n:th expression is present, for $n \in [1, N]$. The reasoning behind allowing several lexical hooks and acronyms to be extracted for each paper is that one expression may not be enough to describe some papers. Consider an example where a paper is cited both for its use of SVMs (Support Vector Machines) and HMMs (Hidden Markov Models). If the acronym "SVM" was chosen to single-handedly represent the paper, all implicit citations containing "HMM" but not "SVM" would receive an acronym-score of 0.

As there is clearly strong relation between the `ACRONYM(i)` and `LEXICAL_HOOK(i)` features in that the acronym "SVM" has identical meaning as the lexical hook "Support Vector Machine", for each extracted lexical hook a corresponding acronym is stored which is also used when checking the presence of the lexical hook in a sentence. So if a sentence contains "SVM" the lexical hook feature matches this to "Support Vector Machine".

A final note on the implementation of features is that many simple tricks were used to ensure that phrases are correctly matched even when slightly distorted, such as some words of a lexical hook not being capitalized (hidden Markov model) or an acronym being tied to other words (HMM-tagger) or being written in plural (HMMs).

Classification

Imbalanced data is a problem in our dataset, since less than 1% of the instances belong to the positive class (implicit citations). There are two common ways to balance the data, namely "up-sampling" (reusing data from the smaller class) and

"down-sampling" (only using a subset of the larger class) [28]. In this work down-sampling is used as the dataset is relatively large, with more than 200,000 sentences. After the down-sampling the classes are balanced.

To prevent the overfitting that may result from the cross validation method used by Athar and Teufel [5], an alternative approach is presented here. Instead of letting Weka construct the folds from the whole dataset, the folds are constructed paper-wise. As the dataset consists of 20 separate cited papers and corresponding citing papers, when classifying one of these, training should be done on the remaining 19. It is therefore a 20-fold cross evaluation. Using the full data would be very computationally expensive, but as down-sampling is used only a subset of the sentences from the 19 papers are used in the training.[2] However, all sentences from the remaining 20th paper are still included in the test-set to make sure the result is credible. This method should result in lower scores for the classifier than the previous approach, as there should be less correlation between sentences from the test-set and training-set, but it may be a more fair and correct evaluation method.

3.3 Graphical approach

In this section the graphical approach is presented. It extends upon the work of Qazvinian and Radev [29] and uses a graphical model and an iterative algorithm to arrive at the sentence classes. Strictly speaking, the output is not a classification but rather probabilities of each sentence being in a specific class. The final classification will depend on a chosen threshold value. Intuitively, setting the threshold to 0.5 makes sense, as that means picking the class that is more likely given the result. However, depending on how you prioritize between precision and recall, other values may be optimal.

Specifically, the method used is based on a Conditional Random Field (CRF) together with an iterative message passing algorithm (Belief Propagation). For an introduction to CRFs, see section 2.3. Below is a description of the algorithm defined by Qazvinian and Radev together with their presented results, followed by the changes made in this work.

3.3.1 Qazvinian and Radev's work

Theoretical background

Each sentence is represented by a hidden and an observed node in the CRF. The observed node represents the known data about the sentence (the text it contains), whereas the hidden nodes represents its class (*not a citation* or *implicit citation*). Formally, the value of the hidden node denotes an assumed probability that the sentence is an implicit citation, and is defined by a *potential function*. The relatedness between two neighbouring sentences is represented by a weighted edge, and

[2] In the down-sampling process, a large number of negative sentences are excluded from each paper, for the training phase.

is defined by a *compatibility function*. It may be more intuitive to reason about the system as if the pair of nodes (hidden and observed) is actually just a single node, as this is more similar to the implementation and the message passing algorithm. Hereafter the focus will be more on implementation details than theoretical justification.

Belief propagation implementation

The method is implemented as a belief propagation algorithm that is described below. The original description from Qazvinian and Radev can be found in [29]. The key concept of the algorithm is that sentences have a "mental state" with beliefs about themselves and each other, and that they influence their neighbours by passing messages. Eventually convergence is reached and the final beliefs of the sentences are assumed to be true.

The event c_i denotes that Sentence i is an implicit citation. Each sentence i has a belief $P_i(c_i)$ about its own probability of being an implicit citation. This belief starts out as $\phi_i(c_i)$, the value of the *potential function* which is essentially a prior probability that is based on the text of the sentence (explained in more detail later). Sentences also have a belief about their neighbours within a specified neighbourhood range. i's belief about j is not static but a function of $P_i(c_i)$ and $sim(i, j)$, the similarity of i and j. The *compatibility function* $\psi_{ij}(c_j|c_i)$ denotes i's belief about j given i's own state. Qazvinian and Radev [29] make the assumption that if a sentence does not believe itself to be an implicit citation, it is indifferent toward its neighbours, or expressed more formally it has no impact on their probability of being implicit citations (Eq. 3.1). However, if it believes the opposite it will consider how similar the neighbours are and take the stance that the more similar ones are more likely to also be implicit citations (Eq. 3.2).

$$\psi_{ij}(c_j|\neg c_i) = 0.5 \tag{3.1}$$

$$\psi_{ij}(c_j|c_i) = S_{ij} \tag{3.2}$$

As $P(c_j) + P(\neg c_j) = 1$ by definition, we also have the trivial formula $\psi_{ij}(\neg c_j) = 1 - \psi_{ij}(c_j)$. S_{ij} is a function of the similarity between i and j. With the reasoning that two very different sentences should not affect each other either (give 0.5 probability), Qazvinian and Radev define this function as

$$S_{ij} = \frac{1}{1 + e^{-cosine(i,j)}} \tag{3.3}$$

where $cosine(i, j)$ is the cosine similarity of i and j, presumably based on the sentences' unigrams.

At each step of the algorithm, every sentence passes messages to its neighbours containing its beliefs about them. Each message contains two values, corresponding to the two classes. As described above, i's belief about j may depend on i's class, the similarity of i and j as well as the compatibility function $\psi_{ij}(c_j|c_i)$. However,

there is one flaw with the description above, which is that sentences do not believe themselves to be either in one class or the other. They have a continuous probability that can range from 0 to 1. This means that both equations 3.1 and 3.2 will be used. The message passed from i to j is defined as

$$m_{ij}(c_j) \leftarrow P(c_i)\psi_{ij}(c_j|c_i) \prod_{k \in ne(i)\setminus j} m_{ki}(c_i) + P(\neg c_i)\psi_{ij}(c_j|\neg c_i) \prod_{k \in ne(i)\setminus j} m_{ki}(\neg c_i) \tag{3.4}$$

$$m_{ij}(\neg c_j) \leftarrow P(c_i)\psi_{ij}(\neg c_j|c_i) \prod_{k \in ne(i)\setminus j} m_{ki}(c_i) + P(\neg c_i)\psi_{ij}(\neg c_j|\neg c_i) \prod_{k \in ne(i)\setminus j} m_{ki}(\neg c_i) \tag{3.5}$$

$ne(i)$ in the formula denotes the neighbours of i. As the messages are all sent simultaneously, m_{ki} denotes messages received in the previous step. These initially have the value 0.5. As the messages contain probabilities, a normalization is performed so that $m_{ij}(c_j) + m_{ij}(\neg c_j) = 1$.

Each sentence then considers the received messages and updates its own belief by multiplying the prior belief with the product of the incoming beliefs. When convergence has been reached, the final belief b_i of a sentence i is defined as

$$b_i(c_i) = k\phi_i(c_i) \prod_{j \in ne(i)} m_{ji}(c_i) \tag{3.6}$$

$$b_i(\neg c_i) = k\phi_i(\neg c_i) \prod_{j \in ne(i)} m_{ji}(\neg c_i) \tag{3.7}$$

where k is a normalization factor ensuring that $b_i(c_i) + b_i(\neg c_i) = 1$. These b_i are then matched against a chosen probability threshold. Sentences whose belief values exceed the threshold are classified as implicit citations and the rest are not. Explicit citations are excluded from the evaluation, as they are assumed to be given in a real use case.

Figure 3.4 shows a tiny made-up example of the algorithm just before the first round of messages are passed. A minimal neighbourhood is used, where only direct neighbours affect each other. "Johnson et. al (95)" is the target cited paper in this example. The prior probabilities are written out in the nodes. The first sentence doesn't seem related to the cited paper in any way, and therefore has a low probability 0.2. The second paper is an explicit citation so the probability is 1. The third sentence contains the term SVM that may be frequently used in reference to the cited paper, so it has a probability of 0.6. Now we consider the neighbour beliefs, written out along the edges. The fact that sentences that are not implicit citations do not affect their neighbours is shown clearly by the multiple $\psi(c_i|\neg c_j) = 0.5$ in the figure. Since the first two sentences are somewhat similar, in that they both contain the word *conclusion* they get a S_{ij} above 0.5. Sentence 2 and 3 on the other hand have no similarity and therefore do not affect each other at all.

The key point about the algorithm is the following. Even though sentence 1 has a low prior probability, if it is similar to another sentence that has a high probability, it may get boosted enough by received messages to still get classified

as *implicit citation*. This makes it possible for the algorithm to correctly identify implicit citations that in isolation may look like they are not citations.

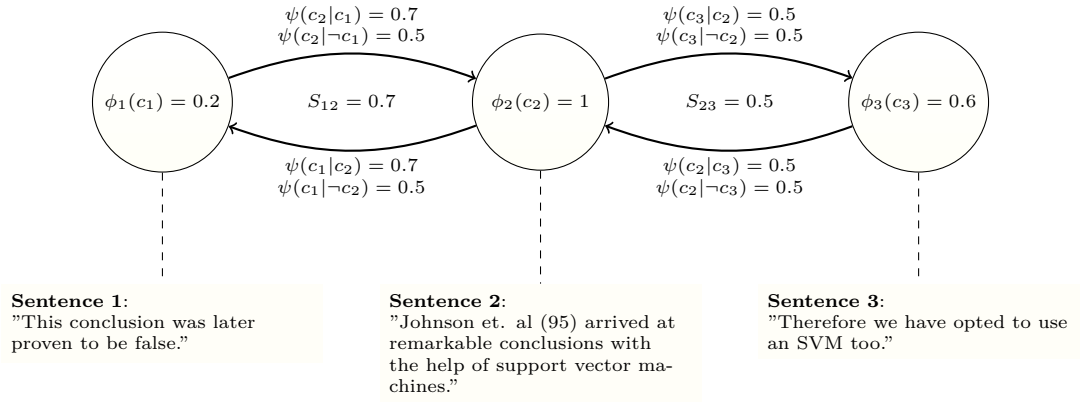


Figure 3.4: Graphical model example with message passing.

In the description above it was stated that the potential function served as a prior probability for a sentence, based on its text. Here follows the formal definition as extracted from the prior work [29]. The potential function of sentence i is computed as a sum of features a, b, c, and then normalized with the rest of the sentences. The features are defined in Table 3.5.

$$sum_i = a_i + b_i + c_i \tag{3.8}$$

$$\phi_i(c_i) = \frac{sum_i - \min_j(sum_j)}{\max_j(sum_j) - \min_j(sum_j)} \tag{3.9}$$

Table 3.5: Potential function features

Feature	**Description**	**Value**
a_i	Sentence i is an explicit citation.	Binary (0 or 1)
b_i	Sentence i contains a determiner from Appendix A.5 followed by a work noun from Appendix A.6 OR Sentence i starts with "this" or "such".	Binary (0 or 1)
c_i	Sentence i's cosine similarity to the content of the cited paper.	Interval (0 to 1)

Qazvinian and Radev's dataset

The dataset of Qazvinian and Radev was constructed in the opposite direction of Athar's dataset. Rather than starting from a set of papers and then adding other papers that cited these, they started from a set of papers and annotated the papers once for each reference. So for one paper there are several annotation instances. In total the dataset consists of 203 such annotation instances.

The gaps between implicit citations and other citations were measured by Qazvinian and Radev and the results are presented in Table 3.6.

Table 3.6: Distribution of gaps between implicit citations as measured by Qazvinian and Radev for their own dataset.

gap size	0	1	2	4	9	10	15	16
instances	273	14	2	1	2	1	1	1

Qazvinian and Radev's results

Qazvinian and Radev's results were only presented as F_3-scores and are quoted in Table 3.7. The most important column to study is $\mathbf{BP_4}$ which shows the result for the algorithm presented above, using a neighbourhood size of 4. $\mathbf{BP_1}$ and $\mathbf{BP_n}$ simply use other neighbourhood sizes. $\mathbf{B_1}$ and $\mathbf{B_2}$ are baselines and **SVM** is a classifier using 4 features (3 of them representing the same data being used in the potential function, and the last one being "distance to the closest explicit citation")

Table 3.7: Qazvinian and Radev's results (F_3-measure)

paper	$\mathbf{B_1}$	$\mathbf{B_2}$	**SVM**	$\mathbf{BP_1}$	$\mathbf{BP_4}$	$\mathbf{BP_n}$
P08-2026	0.441	0.237	0.249	0.470	**0.613**	0.285
N07-1025	0.388	0.102	0.124	0.313	**0.466**	0.138
N07-3002	0.521	0.339	0.232	**0.742**	0.627	0.315
P06-1101	0.125	0.388	0.127	0.649	**0.889**	0.193
P06-1116	0.283	0.104	0.100	0.307	**0.341**	0.130
W06-2933	0.313	0.100	0.176	0.338	**0.413**	0.160
P05-1044	0.225	0.100	0.060	0.172	**0.586**	0.094
P05-1073	0.144	0.100	0.144	0.433	**0.518**	0.171
N03-1003	0.245	0.249	0.126	**0.523**	0.466	0.125
N03-2016	0.100	0.181	0.224	0.439	**0.482**	0.185

One interesting detail about Qazvinian and Radev's evaluation is that they consider explicit citations as negative instances, that is, the algorithm is punished for giving a high probability to these sentences, even though they are given a high initial probability from the potential function. Doing the evaluation this way should result in an even more difficult problem. In this thesis, the explicit citations are

instead excluded from the evaluation since they are trivial to find and are not part of the problem definition.

3.3.2 Contributions of this thesis

Since the data used by Qazvinian and Radev could not be accessed it is hard to reproduce their work. Nonetheless their algorithm was reimplemented and run on Athar's dataset. Inspecting the sentences' cosine similarity to the cited paper revealed that it is often below 0.1 and therefore quite insignificant, considering the other features are binary (0 or 1). It is possible that the similarity values should be normalized so that the values are distributed all the way from 0 to 1. This is tested as well.

We also try borrowing some features from the work of Athar and Teufel to improve the results. For the potential function we use the `LEXICAL_HOOK` and `ACRONYM` features. As the two features may be correlated since the acronym and the lexical hook could refer to the same thing (such as the case of "HMM" and "Hidden Markov Model"), the max value of the features, rather than the sum, is used. Equation 3.10 is then extended to

$$sum_i = a_i + b_i + c_i + d_i \tag{3.10}$$

with $d_i = max(LEXICAL\_HOOK, ACRONYM)$.

For the compatibility function we borrow some features to detect text relatedness, namely `DET_WORK`, `PRONOUN` and `CONNECTOR`. As these features mostly connect sentences to the previous sentence, the relatedness between two sentences S_i and S_j is defined as

$$rel(S_i, S_j) = \begin{cases} 1.0 & \text{if } i = j + 1 \text{ and } i \text{ matches any of the patterns} \\ 1.0 & \text{if } j = i + 1 \text{ and } j \text{ matches any of the patterns} \\ 0.0 & \text{else} \end{cases}$$

and equation 3.3 is then extended to use the relatedness if it exceeds the cosine similarity.

3.4 Similarity measures

For both of the approaches described above, text similarity plays an important role. This text similarity does not need to be limited to cosine similarity of term frequencies. One hypothesis of this thesis is that using a more sophisticated semantic similarity measure may improve the overall results of the algorithms. This section will present several different measures that were tested for this purpose.

In the machine learning approach, three new features related to text similarity were introduced, namely `TITLE_SIMILARITY` (similarity towards the title of the cited paper), `CONTENT_SIMILARITY` (similarity towards the content of the

cited paper), and CITE_SIMILARITY (similarity towards other sentences that are explicit citations).

In the graphical approach, text similarity is used to set the initial probabilities of the nodes, and also when passing messages. There is an issue of normalization here that has to be addressed. The algorithm is constructed with cosine similarity of terms in mind. Other similarity measures may tend to give much higher or lower values. This does not necessarily work well with the algorithm. Different threshold values may be needed or the similarity measures may even have to be adjusted to fall in the interval occupied by the previous measure. This is especially true for similarity measures that may give values above 1. To handle this, the similarities between sentences in the paper are normalized so that their values fall between 0 and a max-value. This max-value is not set to 1 as the cosine similarities very rarely reach that level. Inspection shows that typically the most similar sentences produce a cosine score close to 0.5, and so the new similarities are normalized to the interval $[0, 0.5]$.

Below, we will present the different similarity measures that were used as alternative text representations in the graphical approach.

3.4.1 n-gram variations

Instead of representing the sentences with tf-idf of unigrams, one can of course use higher level n-grams. What is speaking against this idea is that multi-word phrases and expressions are already modelled using the LEXICAL_HOOK feature. Something that is not handled by this feature is skip-grams, which are explained in more detail in chapter 2.2, essentially n-grams where a gap is allowed between the terms. Skip-grams rely less on exact matches when finding similar sentences.

When using longer n-grams than unigrams, one must decide how to combine the different levels of n. In some areas such as language modelling back-off models may be used, which means that one starts out using higher level n-grams but if not enough data is available one backs off to a lower level. It is not entirely clear how this principle can be translated to the problem of text similarity. In some cases the unigram similarity may be much higher than the bigram similarity and vice versa, even if there is enough data for both levels. Therefore, the approach taken in this thesis is to simply take the sum of the unigram similarity and bigram similarity. Since we want the produced similarities to lie in the interval $[0, 1]$ we simply cut off values that surpass 1.

We try using both regular bigrams and the skipgram variant skip-2-bigrams. Texts are compared with cosine similarity of tf-idf scores of their n-grams or skip-grams. For this to be possible we also need idf-values of the n-grams and skipgrams. It would be possible to get these from a separate source, but we simply extract them from all the documents in our dataset. The tf-idf score of an n-gram is computed as

$$tfidf = (1 + log f_{t,d}) * log(1 + \frac{N}{n_t})$$

where $f_{t,d}$ is the frequency of the n-gram in the selected text and $\frac{N}{n_t}$ is the fraction of sentences in the dataset that contains the n-gram.

3.4.2 LSA

The S-Space package [20], an open source Java library from UCLA, was used to model texts as distributional vectors. To build the semantic space, 650 papers from the ACL Anthology Network were used. As the papers were downloaded as PDFs the apache library pdfbox was used to convert them to text. They were then lemmatized using the Stanford CoreNLP toolkit [23], and split into sentences since we are modelling sentences rather than whole documents. The dimensionality of the semantic space was set to 500.

Words are mapped to vectors in the semantic space, so given the space, words can be compared using their vector representation and some vector similarity measure such as cosine similarity. However, it is not completely obvious how to go from this word similarity to sentence similarity. One approach is to simply sum the vectors corresponding to all the sentence's words (or take the average which has the same effect if one uses cosine similarity). However, a problem arises if the word vectors are not centered around the origin in the space. Higgins and Burstein [16] discuss this problem in detail. In general, the longer two sentences are, the higher their vector similarity will be. This is so because the average vector of a sentence, when adding more words to it, will approach the mean vector of the semantic space. The similarity of two sentences that approach the same mean vector, will naturally approach 1. The way this is handled is to subtract the mean vector from each term vector, and thereby make the vectors centered around the origin. A sentence vector is then constructed by averaging these new term vectors. Note that all term vectors are first normalized, both for computing the mean and when constructing a sentence vector.

3.4.3 Wikipedia titles

Simply put, we let a text be represented by all Wikipedia articles whose titles are words found in the text, as well as any other articles that are linked to from these. Specifically, a text is represented by a bag of *concepts*, where a concept corresponds to several Wikipedia articles. For every word in the sentence, if there is an article with that title, a new concept is created. Stop-words are not considered. The concept is made up of that article as well as all other articles that it links to. Cosine similarity is used for comparing texts, with one distinction. Two concepts are counted as equal if there is an overlap in the Wikipedia articles that they consist of. This means that, if two words such as *vertex* and *edge* have corresponding Wikipedia articles that both link to the same article, for instance one about *graph theory*, they will give rise to concepts that are deemed as equal in the cosine similarity comparison.

The Wikipedia titles and links are taken from a dataset compiled by Haselgrove [15]. The dataset was created from a Wikipedia dump in 2009, and consists of 5.7 million articles and 130 million links between articles. The dataset is in the form of two text files containing the articles' links and titles respectively. Table 3.8 shows additional information given about the dataset.

Table 3.8: Information about Haselgrove's Wikipedia dataset

Total number of pages	5,716,808
Total number of links	130,160,392
Max. number of out-links from a single page	5,775
Max. nubmer of in-links to a single page	374,934
Number of pages with no outlinks	10,438
Number of pages with no inlinks	1,942,943

3.5 Combined approach

As we only have two distinct classifiers in this work, using voting may not make much sense. Instead the sentence scores produced by the graphical algorithm are used as features in the machine learning classifier. This means that it is up to the classifier to decide how much impact this score should have in relation to the other features.

Chapter 4

Results

In this chapter, the experiment results are presented. In many cases both an F_1-value and F_3-value are given. Note that in prior work only one or the other was used to present the results. F_3 was presented by Qazvinian and Radev, whereas Athar and Teufel only presented F_1. When nothing else is noted, the F-values in this chapter correspond to the positive class (implicit citations) as it contains the instances of interest.

4.1 Graphical approach

Several versions of the graphical algorithm were run. For all versions the neighbourhood variable was 4, and the belief threshold was 0.5 unless otherwise stated. The neighbourhood size controls how the nodes of the graphical algorithm affect each other and the belief threshold is used at the end of the algorithm to classify sentences according to their final belief values. The notation used for different experiments is explained below.

Naive An initial naive implementation. Similarities used in potential function are not normalized.

Default The default implementation. Like Naive but similarities are normalized to be distributed between 0 and 1.

Default- Default, but without any iterations of the belief propagation algorithm.

Default$_{\mathbf{pot}}$ Default with additional features added to potential function.

Default$_{\mathbf{comp}}$ Default with additional features added to compatibility function.

Ext Extended implementation. Combination of Default$_{\text{pot}}$ and Default$_{\text{comp}}$, all features added.

Ext- Ext, but without any iterations of the belief propagation algorithm.

Table 4.1: Performance for different versions of the graphical algorithm, using a belief threshold of 0.5.

	Naive	Default	Default-	$\text{Default}_{\text{pot}}$	$\text{Default}_{\text{comp}}$	Ext	Ext-
F_1:	.035	.050	.043	**.117**	.055	.116	.111
F_3:	.092	.125	.106	.217	.145	**.235**	.190

It is clear from Table 4.1 that the added features improve the results, especially the potential function features. Combining the features yields roughly the same F_1-value as the potential function features alone, but an increased F_3-value.

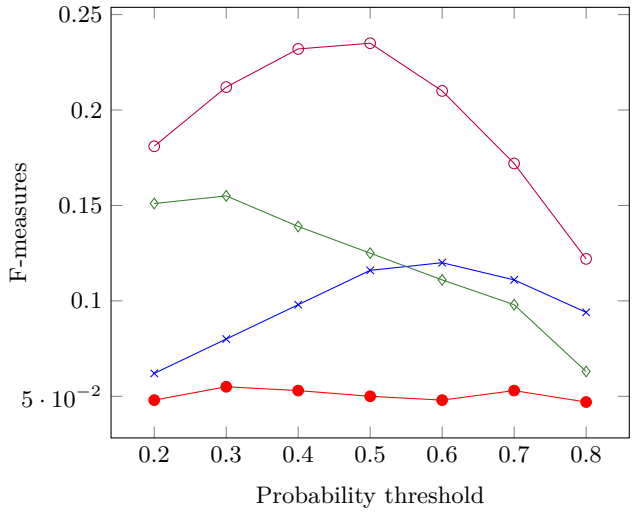
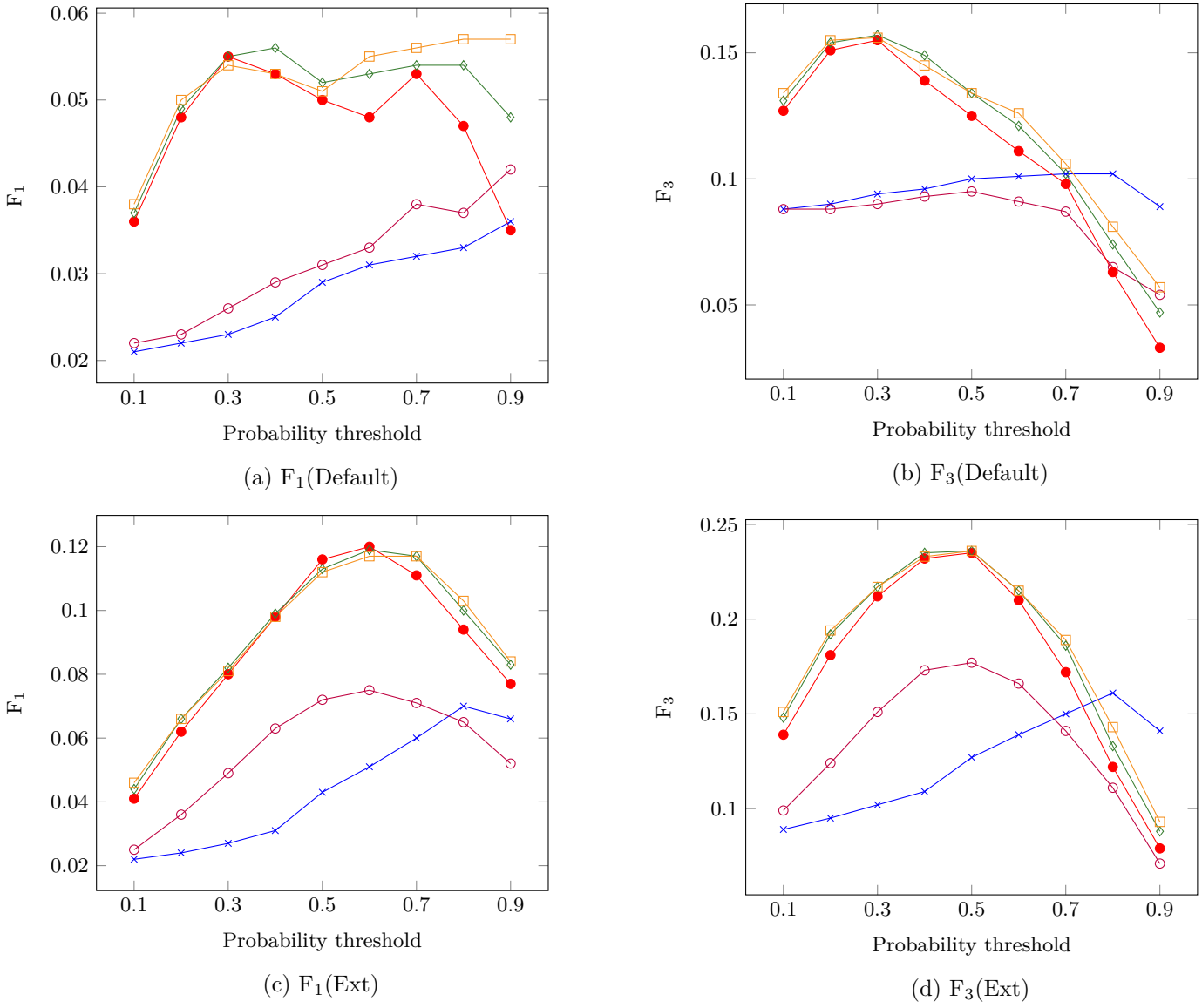


Figure 4.1: Performance of Default and Ext, plotted against different probability thresholds. The probability threshold separates implicit citations from non-citations based on the belief values from the graphical algorithm. F_1(Default), F_3(Default), F_1(Ext), F_3(Ext)

Figure 4.1 shows how the F-values are affected by changing the probability threshold used in the graphical algorithm. It seems that the version without the added features can benefit greatly from a reduced probability threshold, whereas the enhanced version is close to its optimal threshold at 0.5. It is also clear that the F_3-value benefits more from a lowered threshold than F_1.

As the results are clearly very dependent on the chosen threshold value, the results for different similarity measures are also presented this way. Figure 4.2 shows the performance of different similarity measures along with the original approach. The bigram and skipgram measures are quite similar and slightly better than the simple unigram approach. The advantage seems to be less clear when also adding the extra features. The LSA-measure doesn't perform well at all compared to the existing measure. Note that it is of little importance that some measures outper-

form others at a specific threshold level. What matters is really the maximum performance of a measure, since the threshold value is set beforehand.



(a) F_1(Default)

(b) F_3(Default)

(c) F_1(Ext)

(d) F_3(Ext)

Figure 4.2: Performance of different similarity measures plotted against different probability thresholds. The probability threshold separates implicit citations from non-citations based on the belief values from the graphical algorithm. —●— unigram, —□— unigram + bigram, —◇— unigram + skip-2-bigram, —×— LSA, —○— Wikipedia-titles

To get a better understanding of the graphical algorithm, initial belief values of sentences and the rate at which these values changed were also recorded. These

results are presented in Appendix B.

4.2 Acronyms and Lexical hooks

Both the graphical algorithm and supervised classifier make use of acronyms and lexical hooks that are extracted for each of the 20 cited papers in the dataset. Table 4.2 shows the result. For the lexical hooks that consist of several words, the corresponding acronym is also used in the algorithm and shown in parentheses in the table.

Table 4.2: Extracted acronyms and lexical hooks corresponding to the cited papers in the dataset.

paper	**lexical hooks**	**acronyms**
D07-1031	Bayesian	HMM, EM
J96-2004	Kappa	NLP
N06-1020		NANC, WSJ
P04-1015		III, NLP
P05-1045	Gibbs, Stanford	CRF, NE
W02-1011		SVM
W06-1615	Structural Correspondence Learning (SCL), Sentiment Analysis (SA)	III, SCL
A92-1018	Xerox, Markov	SPECIALIST
J90-1003	Mutual Information (MI)	
N03-1003		
P04-1035		NLP
P07-1033		AUGMENT
W04-1013	Basic Elements (BE)	ROUGE, BLEU
C98-2122	Minipar	
J93-1007	-Score, Xtract	
N04-1035		GHKM, SCFG
P02-1053		-IR
P04-1041	Chinese	-II, LFG
P90-1034		
W05-0909	Meteor	METEOR, PER

4.3 Machine Learning - Cross-validation 1

The first results presented are extracted using roughly the same evaluation method as Athar and Teufel, namely performing a cross-validation on the full dataset, without taking the separation of documents into account. However only 4, instead of 10, folds were used.

The results are shown in figures 4.3 and 4.4. Confusion matrices show the exact number of correct and incorrect classifications for both classes. The "positive" class is made up by the implicit citations, and the "negative" class is made up of non-citation sentences. As explicit citations are assumed to be known, they are excluded from the cross-validation. The three classifiers are:

n-grams: Sentences are modelled only by their n-grams.

features: The features from Athar and Teufel are used.

extended: Additional features introduced in this thesis are added.

Table 4.3: Confusion matrices for the classification results using the same evaluation method as Athar and Teufel.

	pred. pos.	pred. neg.
pos.	479	1,280
neg.	209	200,013

(a) **n-grams**

	pred. pos.	pred. neg.
pos.	523	1,236
neg.	173	200,049

(b) **features**

	pred. pos.	pred. neg.
pos.	627	1,132
neg.	183	200,039

(c) **extended**

Table 4.4: Results of the Machine Learning approach using the same evaluation method as Athar and Teufel.

	n-grams	**features**	**extended**
pos. F_1:	.391	.426	**.488**
pos. F_3:	.290	.316	**.377**
neg. F_1:	.996	.996	**.997**

It is clear that the features from Athar and Teufel lead to an improvement over the n-grams baseline, and that the additional features from this thesis further improve the results.

4.4 Information-gain

The information-gain of individual features was measured, using the Weka Java-package. The information gain (IG) of a feature is defined as how much the entropy of the classification is reduced from adding the feature.

$$IG(Class, Feature) = H(Class) - H(Class|Feature)$$

The 15 features with the highest information gain are presented in Table 4.5. In contrast to the information gain presented by Athar and Teufel, here the n-gram features are also presented. It is clear from Table 4.5b that, out of the new features proposed in this thesis, `CITE(i-x)`, measuring the distance to the last previous citation, is the most significant. Also note that the majority of the features with high information gain are n-gram features.

Table 4.5: Information gain of sentence features, measured using the full dataset.

feature	IG $*10^2$
LEXICAL_HOOK(i)	1.03
”roug”	0.82
”meteor”	0.62
CITE(i-1)	0.61
ACRONYM(i)	0.40
”rouge scor”	0.23
”meter”	0.21
AUTHOR(i)	0.19
”rouge-2”	0.19
”scor”	0.18
”scl”	0.17
”mi”	0.15
”rouge-1”	0.14
”bleu”	0.14
”the roug”	0.13

(a) Athar and Teufel's features

feature	IG $*10^2$
LEXICAL_HOOK(i)	1.03
CITE(i-x)	0.95
”roug”	0.82
”meteor”	0.62
CITE(i-1)	0.62
ACRONYM(i)	0.40
”rouge scor”	0.23
”meter”	0.21
AUTHOR(i)	0.19
”rouge-2”	0.19
”scor”	0.18
”scl”	0.17
”mi”	0.15
”rouge-1”	0.14
”bleu”	0.14

(b) extended feature set, with additional features defined in this thesis

4.5 Machine Learning - Cross-validation 2

As pointed out earlier, a more fair evaluation may be to split the dataset according to which paper the sentences belong to. Table 4.6 shows the results using this evaluation method. Here follows an explanation of the classifiers:

n-grams: Sentences are modelled only by their n-grams.

features: The features from Athar and Teufel are used.

extended: Additional features introduced in this thesis are added.

CITE(i-x): Only the `CITE(i-x)` feature is added to Athar and Teufel's features.

combined: The full set of features is extended with one more feature: the score of the sentence given by the graphical algorithm (Extended implementation).

For each fold of the dataset, the performance is displayed for all the classifiers, and the best performing one is marked in bold. As can be seen, the additional features defined in this thesis result in a significantly improved performance. The new `CITE(i-x)` feature results in a large improvement on its own. Adding the scores from the graphical algorithm to the extended feature set slightly improves the F_3-value and decreases the F_1-value. It is also clear that the difficulty of classification is vastly different between different papers.

Table 4.6: Results of the Machine Learning approach, evaluated with the cross-validation method where sentences from one paper do not appear both in the training and test data.

paper	**n-grams**		**features**		**extended**		**CITE(i-x)**		**combined**	
	F_1	F_3	F_1	F_3	F_1	F_3	F_1	F_3	F_1	F_3
D07-1031	.026	.109	.049	.190	**.084**	**.306**	.078	.291	.051	.209
J96-2004	.004	.019	.019	.089	**.025**	**.113**	.024	.108	.023	.105
N06-1020	.022	.091	.081	.275	**.117**	**.367**	.108	.342	.097	.337
P04-1015	.018	.081	.034	.134	.072	.273	**.074**	.274	.073	**.279**
P05-1045	.012	.049	.066	.230	**.100**	**.313**	.083	.267	.085	.298
W02-1011	.016	.074	.043	.178	**.068**	**.258**	.054	.217	.048	.199
W06-1615	.114	.285	.346	.676	**.375**	.685	.378	**.687**	.343	.677
A92-1018	.018	.073	.096	.271	.136	.334	**.137**	.327	.130	**.346**
J90-1003	.033	.127	.059	.237	.076	.288	.072	.277	**.90**	**.315**
N03-1003	.037	.141	.047	.185	.052	.203	.060	.226	**.079**	**.279**
P04-1035	.012	.053	.048	.185	**.067**	**.234**	.060	.215	.064	.230
P07-1033	.030	.118	.056	.168	**.162**	**.366**	.108	.306	.121	.350
W04-1013	.137	.259	.124	.173	.151	.205	.148	.194	**.192**	**.249**
C98-2122	.017	.075	.023	.102	.028	.127	.032	.142	**.042**	**.180**
J93-1007	.022	.093	.034	.145	.047	.195	.041	.174	**.050**	**.205**
N04-1035	.015	.059	.062	.178	.091	.250	.102	.266	**.112**	**.350**
P02-1053	.019	.079	.078	.264	.096	.315	**.107**	**.340**	.088	.299
P04-1041	.010	.040	.055	.198	.069	.243	**.074**	**.257**	.066	.240
P90-1034	.020	.085	.040	.165	.067	.258	.058	.231	**.073**	**.279**
W05-0909	.087	.243	.310	.628	.376	.681	**.384**	.690	.378	**.711**
Average:	.033	.108	.083	.234	**.113**	.301	.109	.292	.110	**.307**

Chapter 5

Discussion

This chapter concludes the thesis. The results from the previous chapter are first discussed in detail, and then summarized in a conclusion. After that we discuss potential error sources as well as generality of the results. Finally future work is proposed and a short discussion of ethics is held.

5.1 Result discussion

5.1.1 Graphical approach

The first results of the graphical algorithm are vastly inferior to those reported by Qazvinian and Radev. There are many possible explanations to this. One is that we used different datasets. There is a rather big variance in the results between different papers, both in the results of this thesis and the prior results. For some papers, the results were almost seven times as good as for others. Thus, it is possible that the dataset used by Qazvinian and Radev may have been easier in a way than Athar's dataset.

The experiments without any iterations were run to see how much impact the iterative algorithm actually has. The results indicate that although it helps, it's not by much, and it's mostly the F_3-value that is increased by the algorithm. This can be explained by the fact that the iterations only increase the probabilities of sentences, meaning that more will be classified as positive. As the F_3-measure values recall higher than precision, it's natural that it goes up.

This was examined further by recording and measuring the initial probabilities and probability changes for all sentences. Due to how the compatibility functions are defined, the probabilities can only be increased from the influence of their neighbours and never decreased. It could be seen that the majority of sentences barely gain any additional probability from the belief propagation algorithm. For sentences that are not citations, this is a good thing. For explicit citations it may be explained by the fact that their probability is already so high from the start. For implicit citations we would want the probabilities to increase as much as possible. They do increase more than for the other classes, but still very few gain more than 0.1. This could

explain why lowering the threshold increased the F-score. If implicit citations rarely gain more than 0.1 probability they would have to start at 0.4 to pass a threshold of 0.5.

Different thresholds were tested. It was shown that especially the naive implementation, without the additional features, benefited from a lower threshold than 0.5. However, one has to be careful of overfitting this parameter.

The features added to the potential function were done so with the motivation that the initial probabilities of implicit citations were too low, even though they are higher than for the negative class. Clearly this had a positive effect, as the results were increased drastically from the addition, confirming Athar and Teufel's findings that the `LEXICAL_HOOK` and `ACRONYM` features are effective indicators of implicit citations.

Shifting focus from the potential function to the compatibility function one can note that the cosine similarity being used is very simplistic, compared to some of the concepts used in the work of Athar and Teufel [5]. If one assumes that the compatibility function is not only interested in the sentences' similarity but also their relatedness, one could look for patterns with the features `DET_WORK`, `PRONOUN` and `CONNECTOR`. It was shown that these features also improved the result, and furthermore, combining the features proved to yield the highest F_3-value.

We draw the conclusion that integrating features from Athar and Teufel to the graphical algorithm yields greatly improved results.

5.1.2 Machine learning approach

For the machine learning approach, two different evaluation methods were used. Both of them clearly show that extending Athar and Teufel's features with a new set of features, defined in this thesis, yields improved results. Especially the distance to the last previous citation proved to be a useful feature, yielding almost as good results as the full extension. This is also motivated by the fact that this feature had the second highest information-gain, measured on the whole dataset, only lower than the `LEXICAL_HOOK` feature. The information-gain experiments also hinted at how grave the overfitting can be when cross evaluating on the whole dataset without splitting on papers, as the majority of the top features were n-grams that do not have anything to do with the concept of an implicit citation, but are rather terms referring directly to some of the papers in the dataset. This further motivates the use of the alternative evaluation method.

The best performance, measured in F_1, was achieved by the extended feature set. Using F_3, the best result was achieved by combining the extended feature set with precomputed results from the graphical algorithm. It is hard to make a direct comparison between the two as there is much variance between different parts of the dataset. Whether or not adding the graphical algorithm score is useful remains somewhat unclear. When comparing the performance to that of the graphical algorithm alone, it seems that the machine learning approach is stronger.

5.1.3 Enhanced text similarity

The results from the graphical algorithm hinted that extending the term-frequency model with higher order n-grams or skipgrams may be useful, but the results were not very significant, especially not when combined with the new feature additions.

Using the distributed word vector approach alone to measure text similarity proved a failure. There are several possible reasons for this. One is that a simple term based similarity measure simply suits the task better. Perhaps the point of the similarity measurement in the algorithm is not to model the topic relatedness of neighbour sentences but rather pick up reoccurring words that link the sentences. The more complex system with a semantic space may simply introduce unneeded complexity. Another possible explanation is that the quality of the semantic space was too low, due to lack of data or a bad choice of dimensionality. The performance-threshold plots showed that this measure required a much higher belief threshold than the others to reach its potential. This means that either the initial probabilities were set too high or the sentences boosted each others' probabilities too much during the message passing phase. Whether or not the distributed word vector approach has potential in this problem domain remains unclear.

5.2 Conclusion

We have shown several points. First of all, the graphical algorithm suggested by Qazvinian and Radev [29] can be improved drastically by adding sentence features from Athar and Teufel. In our experiments, the performance was improved by a **132% increase of $\mathbf{F_1}$** and a **88% increase of $\mathbf{F_3}$** from this addition.

Secondly, the machine learning approach can be improved by adding new features defined in this thesis. Especially the distance to the most recent previous citation sentence proved to be an effective feature. Adding the whole set of new features caused a **36% increase of $\mathbf{F_1}$** and a **29% increase of $\mathbf{F_3}$**.

Combining the methods by adding the score from the graphical algorithm as a feature to the classifier did yield a slightly higher F_3-value, but a lower F_1-value and the differences were not large enough to draw a conclusion from. Comparing the two separate approaches, it seems that the machine learning approach is stronger. It produced a significantly higher F_3-value and just a slightly lower F_1-value, which means that in general the machine learning approach tends to classify more sentences as positive.

Replacing the simple term frequency similarity measure with something more sophisticated did not prove very successful in this work. It seems possible that higher-level n-grams or skipgrams can improve the performance, but after adding other more advanced features this change made a very small difference. The measures based on LSA and Wikipedia-titles both gave weak results compared to the simple term-frequency method.

Even though we produced better results than in prior work, the classifier performance is quite low. Using a measure such as F_3 may make the results seem more

impressive, but it is clear from looking at the F_1-value of .113, produced by the best classifier, that the solution is not suitable in a scenario where one expects reliable classifications. Still, Qazvinian and Radev showed that multidocument summarization systems could be improved by not only using explicit citation sentences but also additional sentences extracted by their system. Similarly Athar and Teufel showed that sentiment detection can be improved by adding these additional sentences. These improvements can be expected to be even more noticable with the changes proposed in this thesis.

5.3 Sources of error

The dataset used was annotated by Awais Athar alone. He annotated 203,803 sentences in approximately 60 hours over a period of a month [4]. This equals 1.1 ($60^3/203,803$) seconds per annotated sentence. Annotating at this speed for roughly two hours per day for a whole month seems like a tough task with a high risk of making mistakes. Some observed problems with the dataset were also brought up in section 3.1, namely faulty sentence delimitation and the inclusion of reference sections. Having the incorrect sentences in the dataset, that are caused by these problems, is bad for two reasons. First of all they may affect the classification of other sentences in a negative way. The supervised classifier draws incorrect conclusions about the classes, and in the graphical model, incorrect beliefs propagate through the network. Secondly, they may deviate from the true patterns in the data and therefore give rise to artificial misclassifications. To make this point clearer, consider a bad "sentence" in the reference section that consists of the title of the targeted reference along with other metadata such as author names and venue. This was annotated as *not citation* even though the sentence contained the full title of the reference. In reality, it seems likely that if someone writes out the full title of another article, it is meant as a citation.

Not everything was crystal clear about the prior works, on which the algorithms were based. One example is how the lexical hooks and acronyms are extracted from explicit citations. As these were two important features the result may be improved simply by extracting more relevant phrases. Although, looking at figure 4.2, it seems that most extracted strings are the kind of phrases you would want.

Down-sampling was used when cross-validating the classifier. It is possible that a better result could be achieved by not excluding any of the data and instead perform some kind of up-sampling.

5.4 Generality

The largest positive finding of this work is that taking the distance to the closest previous citation into account significantly improves the performance of the supervised classifier. Since this was the case for all individual papers of the dataset, it seems likely that it applies in general.

The acronym and lexical hook features introduced by Athar and Teufel proved valuable. But as the features are clearly dependent on the quality of the extracted terms, the question that should be asked is whether or not these features are effective in all datasets. In a research field where acronyms are rare and publications do not typically relate heavily to some specific capitalized phrase, the features would not capture any meaningful information. It is therefore possible that these features can not be applied successfully to all kinds of data, but this has to be answered by future work.

5.5 Future work

Often the problem of finding implicit citations boils down to recognizing that a sentence refers to some entity introduced earlier in an explicit citation. The sentence may refer back to the authors of the other work, or some technique or system presented by them. This is very close to the definition of *coreference resolution*, "the task of finding all expressions that refer to the same entity in a text" [12]. With a reliable coreference resolution system, the relatedness of sentences could be determined by their coreferences and this could be integrated in the graphical model used in this thesis or in some other way. The reason coreference resolution was not explored in this work was that it seemed very computationally expensive and some available systems did not produce satisfying results.

A new set of features was found that improved performance of the supervised classifier. There may be more simple sentence features that can yield further improvements. Most current features focus on one of two things. The first is to judge the likelihood that a sentence is an implicit citation seen in isolation from other sentences. The second is to model its relatedness to previous citations by looking for textual patterns that link sentences together. What is missing is features that look for the opposite, namely changes of topic. If one could reliably detect a topic shift right after an explicit citation this could mean a lowered likelihood of seeing an implicit citation.

The availability of a higher quality dataset with more information could also open up new possibilities. For instance, information about paragraph boundaries could be integrated as a sentence feature, as paragraph boundaries signal topic shifts which may be a useful indicator. Having access to several datasets from different domains would also help determining the generality of this method.

Enhancing the algorithms with more advanced text similarity measures also needs more investigation, before a conclusion can be drawn.

5.6 Ethics

There are some interesting ethical aspects to be considered in the field of AI (Artificial Intelligence) and Machine Learning. First of all, AI usually comes down to the task of automating something that has previously been done by a human. Either it

was done by an end-user in which case this person will just have a simpler experience now, or it was done by a professional. If one considers the latter case, advances in the AI field may result in some people losing their jobs in the long run. However, on a high level this thesis deals with the task of information retrieval, a task that has already moved from experts to the end-users in the last few decades. The other thing one can consider is the data that we feed to the Machine Learning process. It may not be ethical to use any kind of data for all purposes. Our data consists of research papers that are publicly available on AAN. When you publish your research you should expect other people to take part of it and spread it. Furthermore the results we present are not directly related to individual research papers, but rather to the algorithms in use. Therefore there is nothing that indicates an unethical use of the data.

Appendix A

Word lists

This appendix contains the word lists with function words, defined by Athar and Teufel as well as the ones defined by Qazvinian and Radev.

A.1 Athar's determiners

the, this, that, those, these, his, her, their, such, previous, other

A.2 Athar's work nouns

account, algorithm, analysis, approach, application, architecture, characterization, characterisation, component, corpus, corpora, design, evaluation, example, experiment, extension, formalism, formalization, formalisation, formulation, framework, implementation, investigation, machinery, machineries, method, methodology, methodologies, model, module, paper, process, procedure, program, prototype, research, result, strategy, strategies, system, technique, theory, theories, tool, treatment, work

A.3 Athar's connectors

however, although, moreover, therefore, likewise, similarly, also, equally, identically, likewise, furthermore, additionally, first, second, third, firstly, secondly, thirdly, but, still, yet, besides, conversely, otherwise, nevertheless, nonetheless, then, so, lest, indeed, and, such, especially, specifically, particularly, thus, hence, consequently, thereupon, accordingly, even, despite, instead, regardless, whereas, while

A.4 Pronouns

he, his, she, her, hers, they, their, theirs

A.5 Qazvinian and Radev's determiners

this, that, those, these, his, her, their, such, previous

Figure A.1: Same as Appendix A.1 but without *the* and *other*.

A.6 Qazvinian and Radev's work nouns

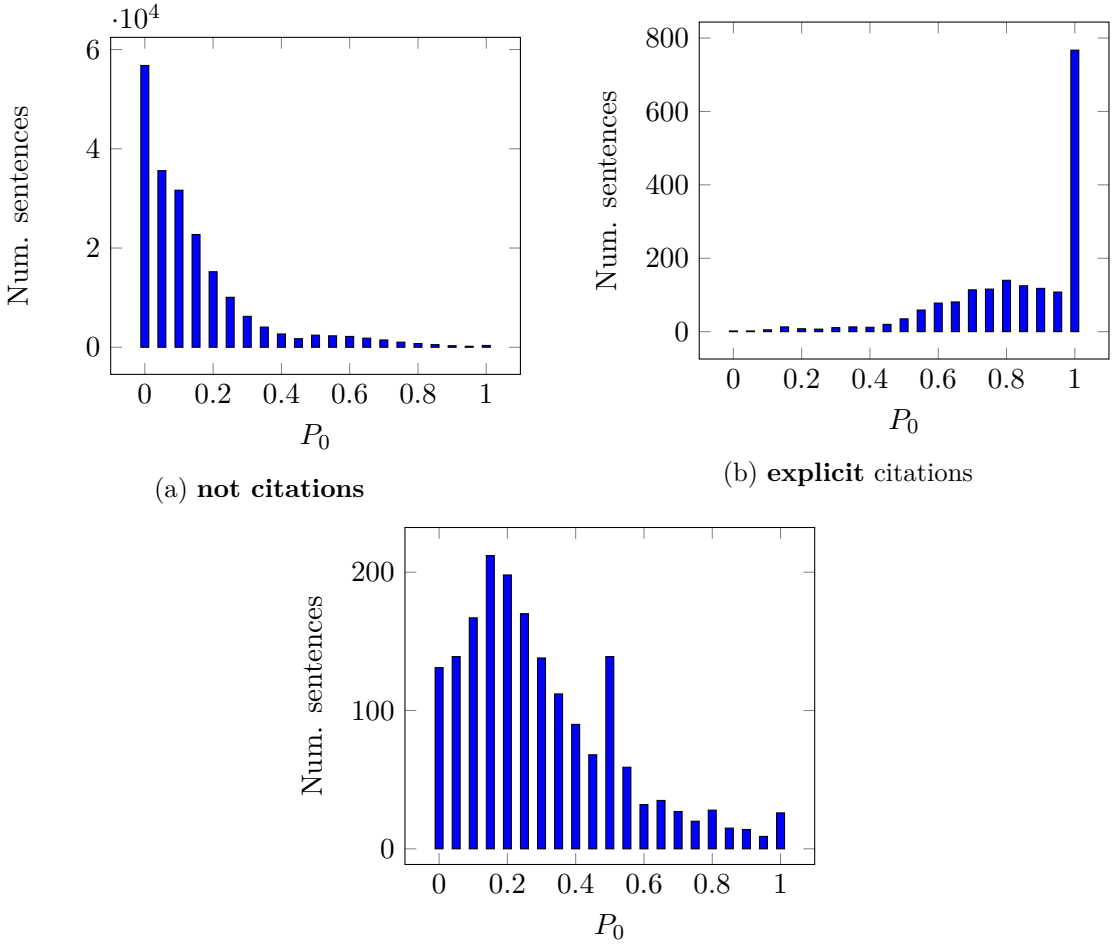
work, approach, system, method, technique, result, example

Figure A.2: A much shorter list than Appendix A.2.

Appendix B

Graphical algorithm results

This appendix contains some results obtained from analysing the graphical algorithm. The purpose of the experiments was to get a better understanding of how the algorithm works, and specifically how the sentence probabilities change during the Belief Propagation iterations. Figure B.1 shows the start probabilities for the different classes in the extended version Ext. The explicit citations receive very high probabilities, the non-citation sentences receive very low probabilities, and the implicit citations are in between but closer to the lower end. Figure B.2 shows how the probabilities of sentences are changed during the iterative algorithm. Implicit citations are increased the most, followed by explicit citations and lastly non-citation sentences. For the majority of sentences, the amount of probability increase is very small.

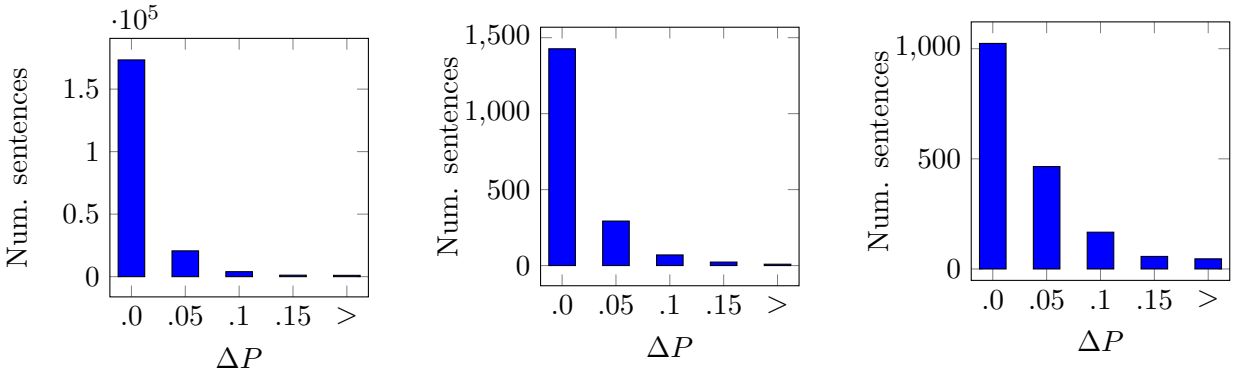


(a) **not citations**

(b) **explicit** citations

(c) **implicit** citations

Figure B.1: Distribution of sentences' **initial probabilities** from potential function (Default implementation).



(a) **not citations**

(b) **explicit** citations

(c) **implicit** citations

Figure B.2: Distribution of sentences' **change of probability** through the belief propagation algorithm (Default implementation).

Bibliography

[1] wikipediaminer. http://wikipedia-miner.cms.waikato.ac.nz/. Accessed: 2015-04-30.

[2] Amjad Abu-Jbara and Dragomir Radev. Reference scope identification in citing sentences. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 80–90. Association for Computational Linguistics, 2012.

[3] Awais Athar. Citation Context Corpus. http://www.cl.cam.ac.uk/~aa496/citation-context-corpus/. Accessed: 2015-05-13.

[4] Awais Athar. Sentiment analysis of scientific citations. Technical Report UCAM-CL-TR-856, University of Cambridge, Computer Laboratory, June 2014.

[5] Awais Athar and Simone Teufel. Detection of implicit citations for sentiment detection. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, pages 18–26. Association for Computational Linguistics, 2012.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[7] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[8] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

[9] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[10] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

[11] Google. Google Scholar. https://scholar.google.se/. Accessed: 2015-05-18.

[12] The Stanford Natural Language Processing Group. Coreference Resolution. http://nlp.stanford.edu/projects/coref.shtml. Accessed: 2015-05-13.

[13] David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4, 2006.

[14] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[15] Henry Haselgrove. Wikipedia page-to-page link database. http://haselgrove.id.au/wikipedia.htm, 2009. Accessed: 2015-05-20.

[16] Derrick Higgins and Jill Burstein. Sentence similarity measures for essay coherence. In *Proceedings of the 7th International Workshop on Computational Semantics*, pages 1–12, 2007.

[17] Jorge E Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences of the United States of America*, 102(46):16569–16572, 2005.

[18] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[19] Mark Thomas Joseph, Rahul Jha, and Dragomir R. Radev. ACL Anthology Network. http://clair.eecs.umich.edu/aan/index.php. Accessed: 2015-05-18.

[20] David Jurgens and Keith Stevens. The s-space package: an open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 30–35. Association for Computational Linguistics, 2010.

[21] John D Lafferty and David M Blei. Correlated topic models. In *Advances in neural information processing systems*, pages 147–154, 2005.

[22] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[23] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[24] Qiaozhu Mei and ChengXiang Zhai. Generating Impact-Based Summaries for Scientific Literature. In *ACL*, volume 8, pages 816–824, 2008.

[25] Michael J Moravcsik and Poovanalingam Murugesan. Some results on the function and quality of citations. *Social studies of science*, 5(1):86–92, 1975.

[26] Jane Morris. Readers' perceptions of lexical cohesion in text. In *Proceedings of the Annual Conference of CAIS/Actes du congrès annuel de l'ACSI*, 2013.

[27] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[28] Foster Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'z2000 workshop on imbalanced data sets*, pages 1–3, 2000.

[29] Vahed Qazvinian and Dragomir R Radev. Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 555–564. Association for Computational Linguistics, 2010.

[30] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[31] Anna Ritchie. *Citation context analysis for information retrieval*. PhD thesis, University of Cambridge, 2009.

[32] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965.

[33] G. Salton and M. McGill. Introduction to modern information retrieval. 1983.

[34] Mikhail V Simkin and Vwani P Roychowdhury. Read before you cite! *arXiv preprint cond-mat/0212043*, 2002.

[35] Simone Teufel, Advaith Siddharthan, and Dan Tidhar. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 103–110. Association for Computational Linguistics, 2006.

[36] Princeton University. Princeton University "About WordNet." WordNet. `http://wordnet.princeton.edu`. Accessed: 2015-04-27.

[37] I Witten and David Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30, 2008.

[38] Xiaodan Zhu, Peter Turney, Daniel Lemire, and André Vellino. Measuring academic influence: Not all citations are equal. *Journal of the Association for Information Science and Technology*, 2014.