Prototype software framework for
causal text mining

L.A. de Vries
School of Management & Governance
University of Twente
The Netherlands
+31 (0) 648951497
l.a.devries@student.utwente.nl

ABSTRACT
Consulting scientific literature in search of relevant cause-and-effect relations is a fundamental part of establishing the validity of new scientific theories within the field of explanatory research, e.g. medicine and social science. The search for literature containing these causal relations is time consuming and often far from exhaustive, leading to duplicate research studies and fragmentation of knowledge. A tool that aids this causal discovery process would therefore enable more exhaustive literature searches and help increase research process efficiency.

This paper describes the development of a comprehensive and extensible prototype software framework, which facilitates the implementation of causal text-mining models and enables indexing of causal relations in scientific documents. One of the available causal text-mining models is implemented within the framework as a proof-of-concept.

Keywords
Causal mining, natural language processing, text mining, software

1. INTRODUCTION
Establishing the validity of a theory by finding and studying relevant scientific literature is a fundamental part of the scientific process, especially in the explanatory field of research. Within this field of research, theories from literature and ‘real world’ measurements are used to describe, explain and possibly predict observed phenomena. It comprises e.g. the natural sciences, medicine and a large part of the social sciences [29,14].

In many cases the search for literature will involve using a predefined search strategy followed by thorough reading of all documents found, in order to be able to ascertain the relevance of these documents to the current research subject [19]. A significant amount of the total allotted research time is, therefore, spent examining scientific documents in search of relevant information. Any efficiency gains in this step of the process will lead to more time being available to other parts within the research process.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post in servers or to redistribute to lists, requires prior specific permission and/or a fee.

12th Twente Student Conference on IT, January 22, 2010, Enschede, The Netherlands.

Copyright 2010, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

In the field of the explanatory research, a literature review is aimed towards finding cause and effect relations in scholarly documents. These causal relations are used to construct a causal model, which describes and explains the observed phenomenon [14]. This model can then be used to design a solution to the described phenomenon. Since creating this causal model is a key objective here, supporting the search for relevant causal relations can lead to efficiency gains in terms of time spent developing the model.

One way of supporting the search process is using a software tool that allows the researcher to model the causal relations in scientific documents in a universal way. The website of TheoryMaps [24,25,12] offers this functionality. It supports the creating of a hierarchy of causal theory maps, and links these modelled theories to the documents they are found in.

Another form of support could come from automating the discovery of causal information in documents. The technique involves automated scanning of a document for causal relations and exporting the resulting causal model to a file or representing it graphically [26]. This allows the researcher to establish if the document has any relevance to the current research without having to read the complete text in search of this information. This paper focuses on the causal discovery process.

The aim is the development of a prototype causal discovery software framework that can be used for implementing the tool described earlier. The framework provides a basis for implementing causal mining models for use within the described tool.

Initial literature review shows the existence of various causal mining models. One of these causal mining models is selected and implemented within the framework to be able to test the framework and to provide basic causal discovery functionality.

To ensure prototype quality, the developed causal discovery software should adhere to certain external and internal requirements, implement an available causal mining technique and have its overall performance measured.

The following research questions address these aspects:

- Which requirements should the developed software satisfy?
- Which causal mining technique should be implemented?
  - Which causal mining techniques are available?
  - How do the various found techniques relate to each other?
3.2 Methodology

Initial exploration of the research topic identified commonly used terminology and keywords. Using these, the following search strings are constructed to search for relevant literature: "causal text mining", "causal knowledge discovery" and "causal knowledge acquisition". Using these search strings, the following process is used to identify relevant articles:

1. An information source is queried. (2) The abstract of the found articles is read to determine which are within the search scope, which is defined earlier this chapter. (3) For each found article a backward and forward citation search is undertaken. If the title of the citation or citing article falls within the search scope, it is examined for relevance. Duplicates of articles that have been found earlier in the search are not examined again. (4) All the found articles are read more thoroughly to establish if indeed they are relevant to the article and within scope.

The found documents on causal mining techniques have been categorised and are displayed in Appendix II.

3.3 Discussion of literature

All found causal mining models are generally founded on the principal of causal discovery by sentence construction and causal cues, i.e. verbs signalling the existence of a causal relation.

The found solutions all use some form of Part of Speech Tagging (POST) [4] to classify the words in a sentence. A POS tagger can determine the lexical structure of a text. One of its functions is distinguishing individual sentences from each other. Another is the tagging of words in sentences according to their grammatical function – e.g. noun, pronoun, verb or adjective. The most frequently used POS tagger in the found solutions is WordNet [23]. This toolkit provides POST and a generic thesaurus – i.e. dictionary – to access additional information on words. One of its features is finding generalisations or specifications of nouns, making grouping of related nouns possible.

The main point of contention within this research field seems to be on whether or not the use of ontology – i.e. real world knowledge of the relationships between words – is required to be able to reliably establish causal relations within texts [30]. On one end of the scientific spectrum Khoo [20,22] and Chang & Choi [16] focus on developing procedures and algorithms that use basic lexical structures to identify causal relations. While on the other end, Chan et al [15] uses domain-specific – thus ontological – knowledge on top of lexical structure identification in order to attain a higher model performance. It should be noted that this domain-specific ontology first has to be constructed [30]. Ontology construction can be done in three ways.

1. Automatically; letting an intelligent computer program learn about the relations within a specific domain.
2. Manually; having a researcher compile and encode the domain relations by hand.
3. Using both techniques; first letting the program compile an initial set of domain knowledge, and then correcting and appending to the set manually.

In all of these cases ontology creation, on itself, can take the scale of a research project. Therefore, the use of ontology for discovering causal knowledge in various different domains is beyond the current research scope.

The previously mentioned causal mining model performance is generally evaluated by measuring the recall
and precision of the model, as the demonstrated by Khoo [21]. Recall represents the percentage of identified causal sentences and precision the amount of indeed correctly identified causal sentences. These two values are combined into a so-called 'F-measure' performance indicator.

Finally, it can be inferred that every new research project, by different authors, starts with creating an implementation of a causal discovery tool from the ground up. This has to occur because none of the studied prototypes’ source code has ever been made available to the scientific community in general. The creation and publication of an extensible causal mining framework is, therefore, instrumental in preventing rework on this basic problem in the future.

4. PROTOTYPE DEVELOPMENT

Each of the subchapters within this chapter specifies a stage within the incremental prototype development process of the prototype. For each stage, the design choices are discussed and a concise description of the work done is given.

The literature review shows that several software tools for textual causal mining have been developed. In all of these cases, however, only the procedural model has been published, not the actual source code. This code will, therefore, have to be recreated.

Furthermore, some of the found mining models are more suitable for implementation within the initial prototype than others. Earlier in this document it had been deemed too time-consuming to synthesise a custom thesaurus or ontology. Thus, solutions relying on this element to function will not be considered for implementation.

The found mining models only deal with the textual processing of a certain text. In order to get to this stage, the scientific texts will first have to be imported into the framework.

Lastly, the prototype is named ‘ÉCRIT’, after its function “Extraction of Causal Relations In Texts”

4.1 Language selection and modelling

Before implementation of the prototype can commence, a programming language is chosen, in which the program will be written. Domain requirements 3.1 and 3.2 constrain the choice of programming languages that may be used for developing the framework.

Study of the TheoryMaps [12] website reveals that it has been implemented using Ruby on Rails [5]. This makes the Ruby programming language [6] the first choice as implementation language, since it satisfies requirement 3.2 – it can be used effortlessly with the TheoryMaps software code. Furthermore Ruby satisfies requirement 3.1, since it is a fully object-oriented (OO) language [7]. Therefore, Ruby is chosen as language for implementing the prototype.

Next, a class model is created for the planned implementation, which specifies the various classes and software methods that, together, make up the prototype. A high-level model is depicted in Figure 1. More detailed information on the inner workings of the software is available through the RubyDoc [16] accompanying the software prototype source code.

![High-level class model](image)

4.2 Program shell

The prototype is constructed in a way that allows it to be run with a command from the command-line or by instantiating the front-end object and calling one of its public methods to load. This empty program shell, which provides an interface for the user and other programs, is constructed first before any functional part of the causal mining process is implemented. This shell provides the required functionality specified in User requirements 2.1 and 2.2.

A user can access the prototype's code via command-line, using:

```
./ecrit.rb [arguments] [input file] [export file]
```

An external program can use the prototype's code by instantiating the 'Ecrit' object, using the following pseudo code:

```
var ecrit = new Ecrit()
ecrit.load("[file_location_string]")
```

The command-line interface provides the basic functionality of inputting a source document and outputting the found causal relations, whereas an external program – directly accessing the framework's classes – can exact more control over the causal discovery process flow and how its results are returned.

4.3 Document import

The first step in the causal discovery process is importing the specified document's contents.

Scientific documents are publicised on the Internet in a myriad of different file formats. However, most scientific documents found, using the Internet search engines described earlier, are encoded in the PDF file format [8]. Furthermore, requirement 1.1 states that the prototype should be able to use PDF documents as input. Therefore, this prototype will initially only provide support for this file format. Support for other file formats can be added to the prototype code during future development.

This functionality is implemented in the DocumentFactory class. The class uses the command-line utility 'pdftotext' [9] to convert the text contained in the PDF file into plaintext, which is returned in a Document object.

4.4 Document pre-processing

At the next step in the process, some pre-processing on the Document object occurs to prepare the text for causal discovery. This step in the process is divided up into two parts: 'base-text extraction' and 'sentence extraction'. The DocumentProcessor class provides the described functionality, for the complete pre-processing step.
4.4.1 Base-text extraction
The first part entails stripping the text body of unwanted parts. These parts are the title, author information and references. These are stored in separate variables within the Document class. What remains is the base-text containing sentences with possible causal information and is stored within a Text object. This object is stored in a variable the Document object.

4.4.2 Sentence extraction
In the second part of pre-processing the Document, the Text object is split into individual Sentence objects using a Part of Speech tagging software tool, thereby fulfilling requirement 1.3.

The likely choice, here, would be to use the WordNet [23] POST tool to split the text, since this tool was found to be the tool of choice in the existing causal discovery models. WordNet, however, depends on three other Ruby packages, two of which proved notoriously hard to install. Therefore, another POST tool, The Stanford Parser [10], is used for sentence splitting. This tool is available in Ruby and only requires Java [11] to function. Java is available on most modern operating systems, thus provides a good alternative to WordNet in this case.

At this point in the development, the document can be loaded and pre-processed. It can be read-out and exported, but nothing else as of yet.

4.5 Causal mining
Next, a causal mining model is implemented to determine the causal relations in the individual sentences.

The implemented causal mining model is part of the method described by Khoo et.al. [9]. This method was chosen, because it is the most basic form of automated causal discovery found. It relies on an exhaustive precompiled list of causative verbs and causal links for discovering causal relations, both of which are discussed below.

Causative verbs are verbs that cause, or bring about, a change. Regard the following sentence: "Susan kills Steve". In this sentence Susan causes Steve to die. The verb thus implies an arbitrary cause, i.e. "Susan fires a gun", and the projected effect: "Steve is dead".

Causal links are words in sentences that indicate a causal relation within that sentence. Regard the following sentence: "Steve got wet because it rained and he forgot his umbrella". In this construction, "[effect] because [cause]", the presence of the word "because" signals the probable existence of a causal relation in that sentence.

While the implementation of Khoo uses both causative verbs and causal links, this prototype only implements the causal links detection. Causative verbs were determined to be too complex to implement within the available research time.

The current model detects causal sentences by overlaying a model sentence construction on a target sentence. The following is an example of this technique:

\{1\} denotes the cause
\{2\} denotes the effect

1) \{1\} because of this, \{2\}
2) because \{1\}, \{2\}
3) \{2\} because \{1\}

These sentences are ranked by how specific they are. (1) is the most specific and highly constrains the numbers of sentences that match it. (2) is less specific than (1) and will therefore match more sentences. (3) is the least specific of the three and matches any sentence with just one word before or after the causative word 'because'. The rule for when getting several matching causal sentences is, to always pick the most precise match i.e. the smallest number. As an illustration consider the following sentence:

"Mary forgot to buy groceries and because of this, she couldn't cook dinner."

This sentence will match with all rules. Matching with (3) reverses the cause and effect, thus an incorrect match. Matching with (2) doesn't specify a cause. Part \{1\} of the sentence just says 'of this'. Matching with (1) yields the right cause \{1\} and effect \{2\} part of the sentence.

This technique was implemented using the sentence construction templates compiled in the extensive work on causation detection by Khoo [20].

Causal relations found using the described model are stored their respective Sentence object.

4.6 Results export
After causal discovery has been performed on the document, it must be returned, displayed or exported to a file. The DocumentExporter class provides this functionality. The class can be called with the Document object to be exported and with a variable specifying how this should be done.

The first option is returning the result as a Document object containing the causal information. In this case, the Document object can be returned directly to the method that called the framework without passing it through the DocumentExporter class first.

Another option is displaying the found causal relations with text. In this case the Document object is fed into the DocumentExporter class and a String is returned using the following syntax:

File: [input_file_name]
Causals discovered:
{[cause]} => {[effect]}
{[cause]} => {[effect]}

The last option stores the found causal relations in a XML file [13]. This file format provides an extensible way of specifying the causal relations found and, furthermore, enables other programs to read-in the result from file. Causal relations are stored using the following syntax:

<document>
<filename> [input_file_name] </filename>
<causals>
<caus>
<cause> [cause] </cause>
<effect> [effect] </effect>
</causal>
</causals>
</document>

5. VERIFICATION
For testing and verification purposes two tests sets of data are constructed. These sets both contain 50 random causal sentences of which 25 came from Internet news sources and 25 came from scientific documents on various topics. The sentences are on a single row of text with one space character separating them.

One set is used for testing newly added or modified code during the development of the causal model, while the other is used as verification set to test the performance of the causal mining model, once development is complete.

Performance is measured in recall and precision for both the discovery of the presence of a causal relation and
establishing the cause and effect part of the sentence correctly.

In the first case recall is defined as the percentage of the total present causal relations the prototype is able to detect. Precision, here, is defined as the percentage of the detected causal relations that is indeed a valid causal relation.

The same definition can be applied to the (correct) detection of cause and effect parts of the causal relation. Table 1 shows the results attained testing the developed model with the verification test set.

<table>
<thead>
<tr>
<th>Table 1: Test results</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall of causal relations</td>
<td>54%</td>
<td>100%</td>
</tr>
<tr>
<td>Cause part of sentence</td>
<td>92%</td>
<td>86%</td>
</tr>
<tr>
<td>Effect part of sentence</td>
<td>94%</td>
<td>78%</td>
</tr>
</tbody>
</table>

The low rate of recall, compared to the original Khoo model, can be attributed to the lack of support for sentences containing causative verbs. These make up 22% of the verification test set. The recall of 54% is, nonetheless, lower than the original model was able to attain. This is probably caused by the structure of the sentences in the verification set.

Once a causal sentence has been correctly identified, the model has a respective 92% and 94% recall of cause and effect parts of the detected relation. In some cases, however, cause and/or effect are not detected at all or are switched.

6. CONCLUSIONS
The developed prototype shows that existing 'in-vitro' developed causal mining models can be implemented within a framework that allows for automated discovery of causal relations in real-life scientific texts. Although its function is currently limited to one causal mining model, it's open and extensible nature allow for further future extension with more models without having to build the framework from scratch.

To answer the first research question, a list of requirements was drawn up. This list specifies the constraints placed upon the prototype by internal and external factors. These requirements can be found in Appendix I.

The second research question was answered by constructing a list of available causal mining models, as seen in Appendix II. The use of ontology or thesaurus in a causal mining model was established as a negative factor in the decision about the inclusion of a model in the prototype. The level of complexity for its implementation was outside de scope of the proof-of-concept prototype. Therefore, it was decided to only include part of the model by Khoo [22] in the developed prototype.

The performance of the prototype was evaluated by use of paragraphs of actual texts containing diverse causal sentences and sentence constructions. These were divided into training and evaluation sets. Evaluation of the final prototype showed a 54% recall of causal relations recall within the evaluation sets overall.

7. FURTHER RESEARCH
The prototype has shown that the implemented causal mining model can detect causal constructions within scientific texts. To improve its performance, future work on this prototype should be focussed on implementing causative verb detection and detecting the polarity of the causal relation. These aspects of the causal mining model were left out of the initial implementation due to time constraints.

Furthermore, the author recommends future studies, involving this prototype or similar prototypes, include process models implemented using a Multi-Agent System (MAS) [17]. Implementing different causal mining models in agents, and letting these agents exchange information to improve the end result may result in a performance increase.

One example of such a system is the agent-based hybrid framework by Zhang et al. [32]. The individual agent’s processing models are static, but they negotiate with each other to form the most optimal output. Another approach [27] let’s the processing models within agents themselves evolve, while the way in which they communicate their results remains static. It should be noted, though, that this last approach is applied to the field of bioinformatics and, more specifically, protein strains. An overview of different agent-based knowledge discovery models can be found in another work by this author [31].

8. REFERENCES
Appendix I: Requirements

The following is a specification of the requirements for the developed causal mining prototype.

1. Functional

1.1 The system should be able to use PDF documents as input.

1.2 The system should output the results in a machine-readable and open format, like XML.

1.3 The system should split the input text into individual sentences.

1.4 The system should be able to use Part of Speech Tagging (PoST) technology to tag words in a sentence.

1.5 The system should implement at least one causal mining model.

1.6 The system should allow for later extension of the causal mining model code by providing the appropriate software hooks.

2. User

2.1 The system should be accessible as a library.

2.2 The system should be accessible via command-line commands.

3. Domain

3.1 The system should be implemented using an Open Source programming or scripting language.

3.2 The delivered prototype should be able to be used with the TheoryMaps software code.

Appendix II: Literature categorised

The following diagram shows the gathered literature on causal mining techniques and the concepts discussed within.

<table>
<thead>
<tr>
<th>Concepts and aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Causation</td>
</tr>
<tr>
<td>Explained</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Articles</th>
<th>Explained</th>
<th>Revealed</th>
<th>Explicit</th>
<th>Specific or generic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khoo 1995</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Khoo 1998</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Khoo 2000</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Chan et al. 2002</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Girju et al. 2002</td>
<td>~</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Girju 2003</td>
<td>~</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Poesio et al. 2004</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Chang et al. 2005</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Tsujii et al. 2005</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Butaria et al. 2007</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>