



NTNU – Trondheim
Norwegian University of
Science and Technology

Using Information Extraction and Text Classification in an Effort to Support Systematic Literature Reviews

Sofien Lazreg

Master of Science in Informatics

Submission date: June 2012

Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Sammendrag

Systematisk litteratur analyse er et viktig verktøy i kunnskap-basert systemutvikling, men krever mye arbeid og tid fra forskere. Ekstrahering av data er et viktig steg i disse analysene, men nåværende praksis krever at forskere manuelt ekstraherer store mengder data. Denne oppgaven undersøker mulighetene for å utvikle en prototype for automatisk ekstrahering, slik at man kan redusere tiden som blir brukt på manuell ekstrahering. Ved å gjennomgå tidligere relatert forskning, og ved å eksperimentere med forskjellige funksjoner og maskinlæringsmodeller ble to forskjellige modeller tilslutt implementert: 'Conditional Random Fields' for informasjonsutvinning og 'Maximum Entropy' for tekstklassifisering. Modellene oppnådde henholdsvis gjennomsnittlig F1 resultat på 67.02% og 73.82%. Disse resultatene kan karakteriseres som gode resultar, og viser at det er fullt mulig å automatisere dataekstraheringsprosessen, ved kun å annotere en liten del av datasettet og for så å lære opp maskinlæringsmodeller til å utføre ekstraheringen.

Preface

This is a master thesis submitted to the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU), in partial fulfilment of the degree of Master of Science. The pre-study for the thesis was conducted partly in Singapore, in collaboration with the National University of Singapore (NUS) and Associative Professor Min-Yen Kan. The thesis was carried out in Trondheim, at NTNU, under the supervision of Associate Professor Heri Ramampiaro and assistance from postdoctoral fellow Daniela Soares Cruzes.

Acknowledgements

I would like to thank my supervisor Heri Ramampiaro, for his patience and assistance, and for all of his advice and guidelines when writing the thesis.

I would also like to thank Min-Yen Kan, and NUS for arranging for my visit and helping me while I learned to use machine learning models.

Special thanks to Daniela Soares Cruzes, who helped me understand systematic literature reviews, and the domain of my thesis.

And finally, I would like to thank my family and my girlfriend for supporting me through the time it took me to complete my thesis.

Trondheim, June 1th 2012

Sofien Lazreg

Abstract

Systematic literature reviews are an important tool in Evidence-based Software Engineering, but require a large amount of effort and time from the researchers. Data extraction is an important step in these reviews, but current practice requires the researchers to manually extract large amounts of data. This thesis investigates the possibility of developing a prototype for automatic extraction, so to reduce the time spent on manually extracting this data. By reviewing related research, and experimenting with different features and machine learning models, two different models were implemented in the prototype: Conditional Random Fields for information extraction and Maximum Entropy for text classification. The models achieved average F1 performance score of 67.02% and 73.82%, respectively. These results can be characterized as good results, and show that it is possible to automate the data extraction process, by annotating a small part of the dataset and training machine learning models to perform the extraction.

Contents

1	Introduction	1
1.1	Background and Motivation	2
1.2	Problem Description	3
1.2.1	Research Questions	3
1.2.2	Problem Definition	3
1.2.3	Scope	4
1.3	Thesis Outline	4
2	Background Research	5
2.1	Introduction	5
2.2	Systematic Literature Review	6
2.2.1	Data Extraction step	7
2.3	Information Retrieval	9
2.4	Information Extraction	9
2.4.1	Machine Learning in IE	12
2.4.2	Evaluation Methods	14
2.5	Text Classification	16
2.5.1	Machine Learning in TC	16
2.5.2	Evaluation Methods	18
2.6	Existing Tools and Development Libraries	18
2.6.1	Stanford Named Entity Recognizer	18
2.6.2	MALLET	18
2.6.3	LingPipe	19
2.6.4	GATE	19
2.6.5	Apache OpenNLP	19

3	Related Work	21
3.1	Introduction	21
3.2	Related Research	22
3.2.1	SVM Approach	22
3.2.2	CRF-based Approach	22
3.2.3	Machine Learning Approach	23
3.2.4	Text Mining Approach	23
3.2.5	Metadata Generation Approach	23
3.3	Existing Systems	24
4	The Approach	27
4.1	Introduction	27
4.2	Training Process	28
4.3	Dataset Preparations	29
4.3.1	Reasoning	29
4.3.2	Collection and annotation	30
4.4	Prototype	31
4.5	CRF component	31
4.5.1	Algorithm details	32
4.5.2	Feature Engineering	33
4.6	MaxEnt component	36
4.7	Preprocessing component	36
4.8	Implementation	37
4.8.1	Resources used	38
4.8.2	System Description	38
5	Evaluation	43
5.1	Introduction	43
5.2	Evaluation Methods	44
5.3	Evaluation Results	45
5.3.1	CRF results	45
5.3.2	MaxEnt results	47
5.4	Discussion	49
6	Conclusion & Future Work	51
6.1	Introduction	51
6.2	Conclusion	52
6.3	Future Work	53

A Dataset	55
A.1 CRF Dataset	56
A.2 MaxEnt Dataset	57
B Annotation Guideline	59
C Lexicon Lists	63
C.1 Countries	63
C.2 Research Methods	64
C.3 Publication Types	64
C.4 Publishers	64
D Evaluation Data	65
D.1 CRF data	65
D.2 MaxEnt data	69

List of Figures

2.1	Graphical structures of machine learning models	14
2.2	K-validation process	15
4.1	Iterative development process for ERA	28
4.2	The ERA system	32
4.3	Different Degrees of State Transition	34
4.4	Example of a sequence of tokens and associated features . . .	35
4.5	Screenshot of ERA	37
4.6	Diagram of ERA back-end system	40
4.7	Diagram of ERA front-end system	41
5.1	Graphical representation of field scores	45
5.2	The different degrees of state transition	46
5.3	Omitting different groups of features	47
5.4	Comparison of different classifiers	48

List of Tables

2.1	Example of a data extraction form used in an SLR	8
2.2	Confusion Matrix	18
3.1	Related research summary	24
4.1	CRF Features	33
5.1	Total number of annotated tokens per field in CRF dataset . .	44
5.2	Total number of annotated tokens per field in MaxEnt dataset	44
5.3	Average per field results of CRF	45
5.4	Average per class results of MaxEnt	47
5.5	MaxEnt confusion matrix	48
A.1	All the papers used as CRF dataset	57
A.2	All the papers used as MaxEnt dataset	58
D.1	Iteration 1 Confusion Matrix	69
D.2	Iteration 2 Confusion Matrix	69
D.3	Trial Confusion Matrix	70

Glossary

CRF - Conditional Random Field

DT - Decision Tree

HMM - Hidden Markov Model

IE - Information Extraction

IR - Information Retrieval

MaxEnt - Maximum Entropy

MEMM - Maximum Entropy Markov Model

NB - Naïve Bayes

Regex - Regular Expressions

SLR - Systematic Literature Review

SVM - Support Vector Machine

TC - Text Classification

Chapter 1

Introduction

1.1 Background and Motivation

As knowledge is becoming more and more accessible on the Internet, the number of available academic material is increasing along with this trend. This increase in available material provides a problem for digital libraries, e.g., ACM¹, IEEE Xplore² and Google Scholar³. The title of academic material, such as papers, are often ambiguous and this causes the libraries to struggle to represent the material in an effective manner. While it has been sufficient to simply return suggested matches to the users information need through query, the users often want additional information to navigate by, such as the year of publication, author of the paper or what type of methodology was used. As a result, the use of information extraction and text classification are heavily researched and is becoming more important when indexing academic material.

Along with this increase in availability, empirical research such as systematic literature reviews, a well-known practice within medical science, has become common in the 'Software Engineering' field. Systematic literature reviews help researchers gather and analyse information provided on the same topic, but from multiple papers. These reviews need to extract a lot of information from the different papers, a practice which is normally done manually by the researchers. Since the dataset often consist of a large number of papers, this becomes a long and time-consuming task. If one could use techniques from information extraction and classification, the same used by digital libraries to represent their material, in an effort to automate this extraction process, one could reduce the time spent on this manual extraction task. The researchers could than continue with the analysis of the gathered information. By annotating a small portion of the dataset, one could try to train machine learning model(s) to extract the information automatically.

Several challenges arise with the task of automating the extraction process. One challenge is that statistical machine learning requires an annotated dataset, which is hard to find in this context. Therefore, a proposed solution would also need an annotated dataset for training and testing. Another challenge is determining what type of information to extract, and if this

¹<http://dl.acm.org/>

²<http://ieeexplore.ieee.org/>

³scholar.google.com

information could be used by every systematic review. By looking at some systematic reviews, one could assess what type of information could be useful to any review.

1.2 Problem Description

The purpose of this section is to give the reader a clear description of the problem. We will state the research questions that will be addressed in this thesis and define the problem more thoroughly. We will also present a scope for the research in our thesis.

1.2.1 Research Questions

The research questions are organized as such: one main, overall research question and three specific subquestions.

Main Question: *Could one develop a method for automatic systematic literature reviews?*

RQ1: *Is it possible to develop a system/tool that could support the data extraction process of a systematic literature review?*

RQ2: *Are there any existing solutions and/or systems that solve the same problems addressed in this thesis?*

RQ3: *How well does the proposed solution work?*

1.2.2 Problem Definition

The problem that the research questions will help us address is the use of state-of-the-art techniques to assist in the extraction of information from academic papers, which are to be used in systematic literature reviews. We need to investigate what type of information would be beneficial to extract, what type of methods have been used in similar tasks and finally develop a proof-of-concept prototype using selected methods. To be able to evaluate our solution, the prototype will require a dataset, and several iterations of development and training.

1.2.3 Scope

The coverage of this thesis will be within the field of software engineering, with focus on techniques from information extraction and text classification, and is limited to the domain of systematic literature reviews in software engineering. The data that will be used by the prototype will consist of software engineering papers, referenced by existing systematic literature reviews. This will ease the dataset collection process, and focus the thesis mainly on the prototype development and on the task of finding suitable methods to perform the extraction itself. It is beyond the scope of this thesis to develop a general system working in other domains. However, with some modifications we believe that our solution will be applicable in other domains.

In view of the above, the main contribution of this research will be insight into how we can help researchers conducting data extraction in SLRs in a more automatic and efficient manner.

1.3 Thesis Outline

The thesis is organized as follows:

- Chapter 1: Introduction to the thesis and problem description.
- Chapter 2: Introduces the background theory for the thesis.
- Chapter 3: Investigates related work and research.
- Chapter 4: Describes in detail our approach.
- Chapter 5: Presents the results, validation data and a discussion.
- Chapter 6: The final conclusion and possible future work.

Chapter 2

Background Research

2.1 Introduction

This chapter will describe the background theory and research needed as a foundation for understanding the rest of the thesis. We will present theory about systematic literature reviews, give a short introduction into information retrieval, navigate through information extraction and text classification, and commonly used methods in both fields, as well as give an overview of available tools and development libraries that could be of use.

The chapter is organized into several sections:

- Section 2.2 Systematic literature reviews theory
- Section 2.3 Introduction into information retrieval
- Section 2.4 Information extraction theory and methods
- Section 2.5 Text classification theory and methods
- Section 2.6 Existing tools and development libraries

2.2 Systematic Literature Review

Systematic Literature Review (SLR) is a research method for reviewing empirical evidence within a field of study, such as Software Engineering. SLRs have been common practice within the domain of medical science, and have later become popular as a tool to support Evidence-based Software Engineering. SLRs are used to help researchers find gaps or links between research, or to provide further evidence that a method/tool is robust by providing a comparison across a wide field of study [1], and is defined as:

“a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest.” [1]

The scientific value of an SLR is dependent on it being thorough, fair [1], and this is supported by the thorough documentation required during the entire review process, and at all its different steps.

An important goal with SLRs is to try and limit any potential bias the researcher has when selecting papers, so to reduce the possibility of selecting only papers that support their initial point of view and argument [2]. This is achieved by following procedures and guidelines that strictly define how to conduct an SLR. A review protocol is defined before the review starts, in the planning phase, and is a detailed plan that specifies the entire process that is to be followed in the review, and any conditions that have been set beforehand [3]. According to [1], there are three main phases to conducting an SLR, with several steps associated with each phase:

Planning the Reivew

- Identification of the need of a review
- Commissioning a review
- Specifying the research question(s)
- Developing a review protocol
- Evaluating the review protocol

Conducting the Review

- Identification of research
- Selection of primary studies

-
- Study quality assessment
 - Data extraction and monitoring
 - Data synthesis

Reporting the Review

- Specifying dissemination mechanisms
- Formatting the main report
- Evaluating the report

All the phases and steps are important when conducting an SLR, but our focus will be on the data extraction step.

2.2.1 Data Extraction step

The data extraction step is conducted by creating forms that are designed to collect all the information needed to address the stated research question(s) [1]. The researchers go through the collected research material and manually fill these forms with the information required. While the research questions are dependent on the SLR, the same metadata fields are quite often extracted across SLRs. These fields are key attributes of the paper, and are non-interpretable. Examples of such fields are author, title, year of publication, and so on. A system that could automatically extract several of these important metadata fields and information from these research papers, could substantially reduce the time spent on the data extraction step, and let the researcher focus on interpreting the data instead.

Table 2.1 shows a partial example of a data extraction form, used in an existing SLR [4]. The 'Fundamental information' and 'Specific information' fields in the form are examples of fields that could be automatically extracted or classified, and can be useful to any SLR. The rest of fields require a researcher to read through the paper and use their judgment and knowledge to fill out, and are SLR specific questions.

Fundamental information

- 1 Data extractor
- 2 Data checker
- 3 Date of data extraction
- 4 Article title
- 5 Authors Name
- 6 Application domain
- 7 Journal/conference/conference proceedings
- 8 Retrieval search query
- 9 Date of publication

Specific information

- 10 Study context
- 11 Research methodology
- 12 Study subjects
- 13 Validity threats

- Academia
- Industry
- Literature review
- Systematic review
- Case study
- Experiment
- Survey
- Action research
- Professional
- Students
- Conclusion validity
- Construct validity
- Internal validity
- External validity

RQ 1 What strategic release planning models have been presented?

- 14 Name of presented model/framework
- 15 Model/Framework proposed in Literature or in industry
- 16 Newly presented model/framework or extension of already developed model/framework
- 17 Means of representation (table, diagrammatically, mathematical means, logically)
- 18 Description of presented model
- 19 On what grounds the model/framework is constructed
- 20 Model or framework use in Industry
- 21 Any requirement selection technique used in the model
- 22 Any limitation of the model/framework
- 23 Practical application of model/framework in the form of tool
- 24 Discussion about any other RP model/framework

RQ 2 What requirements selection factors are discussed?

- 25 What technical and non-technical requirement selection factors are discussed
- 26 Any other name of technical and non-technical requirement selection factors
- 27 Common requirements selection factors discussed in two or more than two models/framework.

Table 2.1: Example of a data extraction form used in an SLR

2.3 Information Retrieval

Information Retrieval (IR) systems are mostly known for their searching ability, where a user states an information need and the system provides the user with a response to this information need in return. IR is a large academic field and encompasses several topics like browsing or filtering documents, processing of retrieved documents and clustering or classifying documents according to their content [5], and is defined as:

“Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).”[5]

A system can increase its preciousness when retrieving the users information need if it indexes good representing facts and features of the text. To find and extract these facts and features, the system can use techniques that specialize in extracting certain proof, or inferred facts, and index these as representation for the documents or text. As the collection of documents expand, automatic techniques that can do this become crucial. Two well-known, and quite different techniques that provide this is Information Extraction and Text Classification. These two techniques could be used together to extract and produce an output that can be representative of a research document, allowing for future IR systems to handle the indexing and retrieval process.

2.4 Information Extraction

Information Extraction (IE) is the process of automatically extracting specific information entities or the relationship between different information entities from a source [6] and presenting it in a structured form. The source of information can be structured like a database, semi-structured like an HTML document or unstructured as free text from an academic paper. We will focus on the unstructured source. IE combines techniques from Natural Language Processing, lexical resources and semantic constraints to effectively provide automatic mining of documents [7] and is one of the most prominent techniques currently used in Text Mining [8].

In [6], the field of IE is presented along these five dimensions:

- (A) The type of structure extracted
- (B) The type of unstructured source
- (C) The type of input resource available
- (D) The method used for extraction
- (E) The output of extraction

A. Type of structure extracted

An IE system can extract several different types of structure but the two most common are entities and the relationships between entities. Entity extraction is used to identify and extract specific entities from the unstructured source. An example of this is Named Entity Recognition (NER), which can be used to extract names of companies, people, locations or identify proteins in biomedical texts. Its area of use is still heavily researched. Relationship extraction refers to the task of identifying and extracting relationships between entities in the unstructured source. This relationship can for example be whether an entity is a subset of another entity or is located in the same proximity. Extracting the relationship between entities differs from extracting just the entities, as it requires the recognition of an association between two different tokens or text segments [6] which is often not stated explicitly in the source.

Named entity and relationship extracting could be useful in our research by extracting and filling out metadata fields like the publisher or the title of the paper. Examples of such metadata fields can be seen in Table 2.1.

B. Type of unstructured source

The unstructured source for an IE system can be full-length documents, segment of the documents, parts of the text, sentences or records. This choice is largely dependent on the application area of the IE system. Academic papers contain a certain degree of structure, making it a partially structured domain specific source [6]. This structure is present in the setup of academic papers, with predefined and well known sections like for example abstract,

introduction, method, result and discussion. There is also a segmentation called “header”, where all the text from the beginning of the paper and up to either the introduction section or the end of the first page is used as source [9].

C. Type of input

The type of input the IE system is provided with can significantly aid the extraction process [6]. A labeled dataset is an example of an input resource where a corpus has been annotated by human(s), according to a predefined annotation guideline. This labeled, unstructured source provides contextual information about an entity making it very valuable for the IE system [6]. The input can also be preprocessed, a technique that can help to identify certain patterns in the natural language. The preprocessing is typically done by sending the unstructured text through a series of “pipes”, either at token-level or sentence-level. Each pipe is programmed to look for certain clues in the input and add labels to the text with this identified information or alter the text. Examples of such pipes are POS, part of speech taggers which tag each word with its grammatical category, or parsers that can alter the tokens into only lowercase characters or remove all non-alphabetical characters.

D. Extraction methods

Different methods can be used for the extraction process itself, and even combinations of these methods. These methods can be hand-coded/rule-based methods, requiring a domain expert and programmer to specify rules and regular expressions to perform the extraction, or learning-based/statistical methods, which rely on manually annotated data to train machine learning models [6]. We will focus on the learning-based method, using machine learning models to perform the extraction.

E. Output

The type of output the IE system will finally produce depends on its area of use. Entity relationships could be presented to the user in a graphical way, such as a tree or graph. If only metadata is extracted, a database can be created with the fields or the fields can be presented in a structured, template format to a user.

2.4.1 Machine Learning in IE

The use of machine learning models to perform sequence labeling is very common in fields such as Artificial Intelligence and Natural Language Processing. In supervised machine learning, the model is given a labeled set of examples. It is then trained to statistically recognize patterns in new documents, assigning labels to fields which the model detects as relevant. To improve these models, a set of features can be stated to help the system recognize the different patterns. This is called feature engineering, and these features can range from simple regular expressions that identify fields like email or phone number, dictionary features that identify membership to a lexicon or token features that tell whether the token contains only lowercase or if it is hyphenated.

We have briefly introduced a series of machine learning models that have previously been used in IE.

Hidden Markov Models

Hidden Markov models (HMM) are an extension of Markov models, where each transition from an observable state has an associated probability and produces a specific output when occurring. With HMM on the other hand, each state is hidden and only has an associated probability with its stochastic state transition that generates an observable output [10]. HMM has been successfully applied to IE [11], NER [12] as well as speech recognition [10].

The advantages with using HMM are the availability and easy-to-use frameworks that exist. These frameworks allow for training by maximum likelihood, using labeled data and calculating the Viterbi path to find the most likely hidden sequence of state(s) [13]. The disadvantage with HMM is that it is a generative model that makes an independence assumption about its observations, limiting the knowledge that previous and future observations provide [14].

Maximum Entropy Markov Models

Maximum entropy Markov model (MEMM) is a combination of HMM and maximum entropy model. In MEMM, state transitions depend on non-

independent features which means that previous and future observations can be taken into account when calculating the probability of a transition. A single function provides the probability of the current state given the previous state and the current observation [13].

The advantage with using MEMM's is the use of non-independent features, which allow for richer representation of the observation sequence at different levels of granularity [14]. MEMM's major disadvantage is the label bias problem. This problem occurs because of a bias towards states with few outgoing transitions. Since transitions leaving a state comply only against each other, states with one outgoing transition have to take this route no matter what their observation is [14]. This means that the label is chosen because it is available, and not the best choice.

Conditional Random Fields

Conditional random fields (CRFs) are discriminative undirected trained models that specifies the probability of a label sequence, given an observed sequence [14]. CRFs use of arbitrary features and labels from previous and future tokens provide a powerful framework for sequence labeling [6], and this has made the model popular to implement in a variety of different uses, such as finding protein names in text [15] and extracting information from papers [9].

CRF offer several strong advantages over HMM and MEMM, like its ability to relax the independence assumptions made in HMM and other generative models. Another advantage is CRFs avoidance of the label bias problem that MEMM suffers from [14], and its ability to obtain a globally optimal label [16]. These advantages and published experimental results in [9, 14] have made CRF the current state-of-the-art method used in sequence labeling [6]. The disadvantages with CRF is its slow convergence during the training process compared to HMM and MEMM [14]. This can cause the time required to train the CRF on a large dataset to be long.

Figure 2.1 shows the graphical structure of the different models that have been introduced. The non-grey circles indicate that the state is not generated by the model.

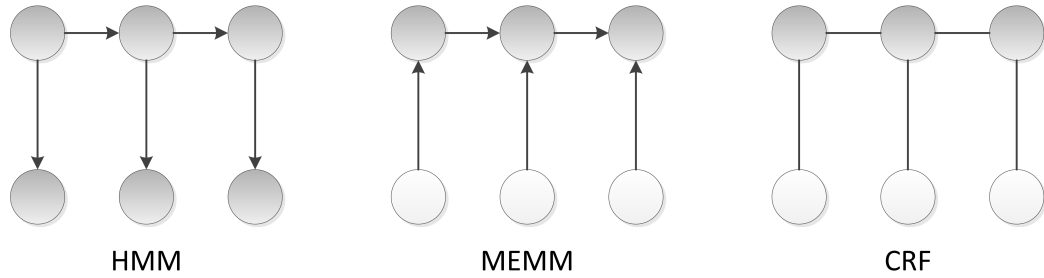


Figure 2.1: Graphical structures of machine learning models

2.4.2 Evaluation Methods

To evaluate the performance of an IE system it is common to measure three metrics used in IR: precision, recall and F-measure.

In IR, precision is the fraction of retrieved items that are relevant [5]. In IE, it would be the fraction of extracted items that were correctly extracted, as presented in Equation 2.1. A high value of precision tells us that the system returned more correctly extracted items than non-correct.

$$P = \text{correctly extracted items} / \text{extracted items} \quad (2.1)$$

In IR, recall is the fraction of relevant items that are retrieved [5]. In IE, it would be the fraction of correctly extracted items that were returned, as presented in Equation 2.2. A high value of recall tells us that the system returned a lot of the correct items available.

$$R = \text{correctly extracted items} / \text{total correct items} \quad (2.2)$$

F-measure is the weighted harmonic mean of precision and recall [5]. F-measure provides us with a single measurement of the effectiveness of the system, with respect to precision and recall. Equation 2.3 presents how to calculate the balanced F-measure.

$$F1 = 2PR/P + R \quad (2.3)$$

It is also common to perform a k-fold cross-validation, where the dataset is randomly split into k subsets and each subset acts as the validation data once and the rest of the subsets act as training data [17], as illustrated in Figure 2.2.

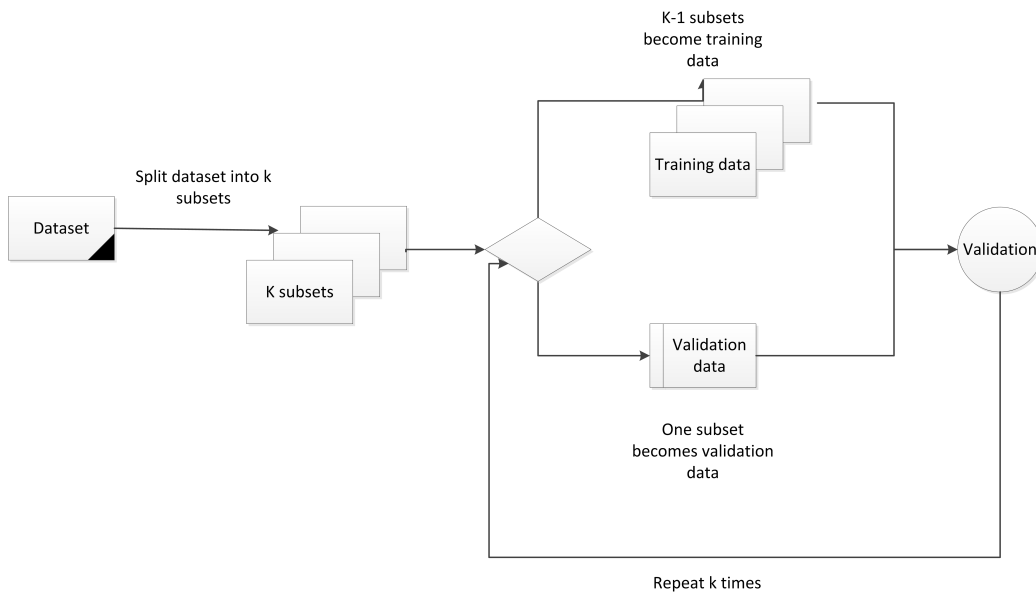


Figure 2.2: K-validation process

2.5 Text Classification

Text Classification (TC), or Text Categorization, is the task of assigning a text, which can be segments or entire documents, into a category or class. As with IE, TC can be performed by predefining rules or using statistical machine learning. TC has been used to automatically detect spam pages, create topic-specific search engines, and classify movie or book reviews into either a positive or negative class [5].

It is important to distinct between the difference of IE and TC, as it can easily be mixed together. While both techniques use machine learning methods, the difference is that in TC the goal is to find the best class for the entire document, text or instance [5], while in IE the goal is to extract specific pieces of information or relationships.

2.5.1 Machine Learning in TC

Using machine learning in TC reduces the need for manual classification, although it requires an initial labeled dataset as discussed in Section 2.4.1.

We have shortly introduced three common methods used in TC.

Decision Tree

A decision tree (DT) consists of a collection of decision nodes that are connected by branches, extending from the root node to the leaf node. The decisions in the decision nodes can be trained from a labeled dataset. Two common algorithms for generating decision trees are CART, classification and regression trees, and C4.5 . CART produces binary decision trees, with only two branches for each decision node. C4.5 produces a more variable tree as it is not restricted to binary splits, and uses entropy reduction to chose the optimal split when recursively traversing the tree [18]. DT has been used heavily in data mining, where the decision nodes are replaced with data, as well as in machine learning.

The advantage with using decision tree is its flexibility, and its capability to break down the classification problem into several simple decision steps making it easier to interpret [19]. The disadvantage is that creating an optimal decision tree is an NP-complete problem, and to overcome this, greedy algorithms have been used to find local optimal decisions instead of global

optimal [20].

Naïve Bayes

Naïve Bayes (NB) is a probabilistic method with positional independence assumptions. There are two common NB models, namely the multinomial NB model and the multivariate Bernoulli model. The multinomial model uses a bag-of-words representation, where the sequence of terms that occur in the document represents it. The multivariate model uses a binary vector to indicate whether a term occurs in the document or not. Both models use a vocabulary to decide which terms to include and exclude [5]. NB uses MAP, maximum a posterior, to decide which class is the most likely to represent the document.

The advantages with using NB is that the multinomial model take document length into account, while multivariate is computational effective and requires only occurrence of token [21]. NB's simplistic nature and easy-to-implement framework are some of the reasons to why it has become such a popular machine learning method [22]. The disadvantages with using the different models are that the multinomial model assumes independence between words and multiple occurrences of the same words, while the multivariate model ignores frequency of tokens and ignores document length [21].

Maximum Entropy

Maximum Entropy (MaxEnt) is a technique that estimates the probability distribution of a class given a document by deriving a set of constraints from a labeled dataset and its features, and using these as constraints for the model distribution [23]. The model that has the highest probability distribution after considering the constraints, is chosen.

An important advantage with using MaxEnt over Naïve Bayes is that it does not make any independence assumption [23]. A disadvantage with using basic MaxEnt is its overfitting problem, which is a problem for all learning models. This problem occurs when there is little training data and poor features to represent it. Several implementations of MaxEnt reduce this problem by introduction maximum a posterior, and a Gaussian prior [23].

2.5.2 Evaluation Methods

TC shares the same evaluation methods as IE, as described in Section 2.4.2, with the addition of using a confusion matrix. This matrix is an important tool when analyzing the classification, and which part of the system might need improvement [5]. As seen in Table 2.2, the matrix shows which classes were mislabeled, and what they were mislabeled as. With this information, one could add features to distinguish more significantly between the misinterpreted classes.

		Predicted	
		Label X	Label Y
Actual	Label X	true positives	false negatives
	Label Y	false positives	true negatives

Table 2.2: Confusion Matrix

2.6 Existing Tools and Development Libraries

There are several existing IE and TC tools and development libraries available which can be used to assist with our prototype. Some of these are introduced below.

2.6.1 Stanford Named Entity Recognizer

The Stanford NER (Named Entity Recognizer)¹ is a CRF-based tool that provides the possibility of labeling entities found in a text, such as persons and company names or protein names. Their webpage also provides several models that have been trained to extract a variety of different classes.

2.6.2 MALLET

MALLET (MACHINE Learning for Language Toolkit)² is a Java API that provides tools for natural language processing, information extraction, classification, sequence tagging and topic modeling. There is a wide variety of

¹<http://nlp.stanford.edu/software/CRF-NER.shtml>

²<http://mallet.cs.umass.edu/>

implemented algorithms available, which include HMM, CRF, DT and NB. The API also provides tools for transforming and handling text into numerical representation and preprocessing options.

2.6.3 LingPipe

LingPipe³ is a tool kit, where a Java API is provided that supports a large variety of linguistic computational operations, such as Part-Of-Speech tagging, spelling correction and sentence detection.

2.6.4 GATE

GATE⁴ is an open-source collection of tools for text processing, used both commercially and scientifically. The GATE Developer provides graphical interactive tools for processing natural language text.

2.6.5 Apache OpenNLP

Apache OpenNLP⁵ is a toolkit for processing natural language text that support tasks such as entity extraction, parsing, chunking and conference resolution.

³<http://alias-i.com/lingpipe/>

⁴<http://gate.ac.uk/>

⁵<http://incubator.apache.org/opennlp/>

Chapter 3

Related Work

3.1 Introduction

This chapter will present research that is related to our thesis, both in the form of related research and existing systems, and mention how our research differs. We will also explain why the existing systems cannot be used in our domain.

The chapter is organized into several sections:

- Section 3.2 Related research
- Section 3.3 Existing systems

3.2 Related Research

The related research into IE and TC is comprehensive, but several articles were found to be of particular relevance. The papers are presented in chronological order.

3.2.1 SVM Approach

Lee Giles et al. [24] presented a paper where they discussed the use of machine learning to automatically extract metadata from research papers, and presented a new method of extracting this data. The method uses, among other things, a priori information of the structural patterns of the data with a Support Vector Machine (SVM) for the metadata extraction. The SVM classifier presents promising results over the HMM methods.

The authors provides us with knowledge about structural patterns, and their possible use in metadata extraction, but differs from our research in that we will be looking at the use of IE in a different domain, to serve as a proof-of-concept.

3.2.2 CRF-based Approach

McCullum and Peng [9] presented a paper where they investigated the use of IE from research papers, using the machine learning method CRF. They experimented with the use of different prior distributions for regularization to avoid overfitting and presented the effects of different features used, both state transition features and local/layout features. They obtained and presented state-of-the-art performance when using CRF to extract fields from research papers.

This research provides us with valuable information regarding the use of CRF in IE, but differs from ours because we are looking at the use of IE within the domain of SLRs, while they have focused on achieving the best possible result from a machine learning model.

3.2.3 Machine Learning Approach

Hu [25] presented a paper where a machine learning approach to automatically extracting the titles of general documents, such as Word and PowerPoint documents, was proposed. The paper concentrated on the use of format features with machine learning models. They tried several different models, and they found that the performance of the CRF model presented the best results. They concluded with the fact that extracted titles can improve the precision of document retrieval.

Hu's research provides us with yet another case of successful implementation of CRF, but differs from our research in that they have focused on automatically extraction titles, while we want to extract several different fields.

3.2.4 Text Mining Approach

Ghani et al. [26] presented a paper where they investigated the use of text mining to extract semantic and explicit attributes for products, which can be used to enhance product databases for retailers. They used a semi-supervised approach, where an initial unsupervised algorithm extracts seeds, or labels, that is then used as training data to a supervised NB classifier and a semi-supervised co-EM classifier. They presented promising results, and have created several prototypes that are based on this initial system.

While this research provides us with knowledge about mining semantic and explicit attributes, and the use of semi-supervised approach, our research will focus on extracting several specific metadata fields, some which might need to be inferred using classification.

3.2.5 Metadata Generation Approach

Lu et al. [27] presented a paper where they investigated automatic generation of metadata from OCRed (Optical Character Recognition) articles. They used an SVM Light Classifier as the machine learning method, and integrated the system into an existing digital library where they achieved promising results.

This paper provided us with the knowledge that OCR could be used to retrieve data from articles, as a step in automatic generation of metadata. Our research will focus on more direct automatic extraction from uploaded papers, but this research provides knowledge into the possible use of OCR as a step to extract the text from the papers.

Article	Approach	Method
[24]	IE	SVM
[9]	IE	CRF
[25]	IE	Several
[26]	Text Mining	NB
[27]	IE	SVM

Table 3.1: Related research summary

The methods presented by the articles allow us to see what is the current state-of-the-art that is used in similar cases as our thesis. This gives us a vantage point when selecting methods to use in our own approach. Table 3.1 presents an overview of all the relevant articles presented in this section.

3.3 Existing Systems

There are several existing systems that have integrated some form of IE and/or TC and these are often digital libraries. The digital libraries use this information to index academic papers, so that they can offer improved and pinpointed retrieval.

CiteseerX

CiteseerX¹ [28] is a digital scientific library that provides a wide range of options, such as automatic reference linking, query-sensitive summaries and automatic extraction of metadata from their indexed papers. CiteseerX actively crawls and harvests papers from digital academic publishers from primarily the field of computer science. While some metadata fields are extracted, e.g.,

¹<http://citeseerx.ist.psu.edu/>

author, title and publication venue, it is still not enough to be used in a typical SLR. The collection of papers is also not complete, which can provide difficulties for researchers when conducting SLRs.

Textpresso

Textpresso ² [29] is an open-source text mining system for academic literature that provides full text searches, TC and mining from biomedical papers. Textpresso can also split entire papers into individual sentences and sort these into semantic categories. It has been implemented into a variety of biomedical systems. and could possibly be used to assist in a biomedical SLR, but has not yet been implemented into the software engineering field.

ABNER

ABNER ³ [15] is a named entity recognition tool that can analyze biomedical text and identify several named entities. The software uses linear-chained CRFs, and includes two models that were trained against two different corpora. ABNER is not trained to process software engineering papers, and would therefore not work as a support tool for SLRs within that domain.

²<http://www.textpresso.org/>

³<http://pages.cs.wisc.edu/~bsettles/abner/>

Chapter 4

The Approach

4.1 Introduction

This chapter will describe the details of our prototype, and the reasoning behind the choices we made. We will describe the preparations that had to be made to the dataset, such as the annotation process that was used and the different fields that were extracted. We will give a detailed explanation of the two classifiers and their implementation, and present ERA, a system that implements the trained classifiers.

The chapter is organized into several sections:

- Section 4.2 Introduces the training process
- Section 4.3: Describes the preparations and collection of the dataset
- Section 4.4 Introduces our prototype
- Section 4.5: Explains the CRF component
- Section 4.6: Explains the MaxEnt component
- Section 4.7: Explains the preprocessor component
- Section 4.8: Describes the implementation details of ERA

4.2 Training Process

The idea was to create a proof-of-concept prototype, that could extract and classify relevant fields automatically using IE and TC techniques. To complete this task, we had to undergo several steps. First, we had to identify what type of information we wanted to extract, and how this could be extracted from the dataset. Second, a dataset was prepared for the training of the machine learners and an annotation guideline was created to handle the annotation process. Third, two machine learning models were developed. Finally, the dataset was used to train the two machine learning models. IE was used to extract most of the fields except in one case, where TC was found to be more suitable for the task, which is discussed later in this chapter.

This process went over several iterations, with each iteration aiming to expand the size of the training dataset with more annotated data and add features to the machine learning models. The goal with each iteration was that the machine learning models would become more advanced, and be trained against an increased dataset. This iterative process is demonstrated in Figure 4.1. After each testing phase, experimental trials were run to see which fields produced poor results. Features were then created or altered, either to specifically boost the poor results produced from a single field, or to boost overall performance of the prototype.

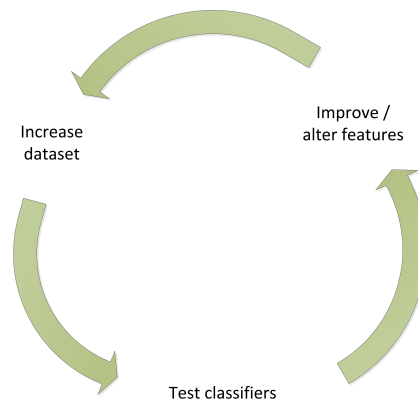


Figure 4.1: Iterative development process for ERA

4.3 Dataset Preparations

Since we did not have an annotated dataset for our prototype, we needed to manually create one. This required us to identify what type of information researchers needed extracted when conducting a systematic review. After reviewing the data extraction process conducted in a couple of software engineering systematic reviews [30, 4], we noticed that metadata fields were quite commonly extracted as well as some key specific characteristics like what type of study context or research method was used in the data. This led us to several fields that could be useful for researchers when conducting a systematic review. These fields are:

Title

Research Approach

Research Method

Publisher

Year of Publication

Publication Type

Country of Origin

4.3.1 Reasoning

A research papers' title is a descriptive sentence, which tells us about the papers' topic area and often contain valuable information about different methods used in the paper. An extracted research title could be used by the researchers to get an overview of the papers it represents, and tell the researcher about part of its content.

A papers' research approach is the path the researcher(s) takes to produce the knowledge the paper presents. There are three main types of research approach:

Exploratory: If the research investigates a problem that has not been defined, and helps to determine the best research way or design, the best data collection method and the best method for selecting subjects [30].

Empirical: Whether the findings in the research are from direct or indirect

observations, such as a case studies [30].

Descriptive: If a system, tool or method is presented and described in the research [30].

A research method is the specific method the researchers use to answer their research question(s). Since there is a wide variety of research methods used in software engineering, the extraction was limited to three common empirical research methods:

Case Study: A case study focuses on what is happening in a single project [31].

Experiment: An experiment focuses on a controlled project, which is replicated several times with random variables [31].

Survey: A survey focuses on knowledge gathered from multiple projects, and tries to capture the bigger picture [31].

The rest of the fields can provide the researcher with valuable metadata information. These metadata fields can be used to infer other type of information, such as which country produced the most research within a specific subject, at a given year or what the most common publication type for that subject is.

4.3.2 Collection and annotation

Several different academic software engineering papers were collected from two different systematic reviews [30, 4] and since the dataset needed to be manually annotated, an annotation guideline was created. This guideline was used during the annotation process to ensure consistency and uniformity of the labeling task. Only the first page from each paper were extracted. One of the reasons for this limitation was that if we were to use the entire paper, the machine learning models would be trained on a larger set of noise text, meaning all the text that is not a part of the fields we want to extract. This abundance of noise text could cause the statistical machine learning model to be poorly trained to recognize the fields. If we limited the text

used even further, such as only using the abstract part of the paper, not enough information would be present to annotate several of the fields and the machine learning model would lack the statistical data needed during training to be able to find these fields.

These first pages were originally extracted using OCR (Optical Character Recognition) software, but this proved to cause some difficulties with handling of the text in the preprocessor. The simplest solution to avoid this problem was to manually copy and paste the first page from the paper into a standard text file, and adjust some of the structure.

The final training dataset consisted of 38 different papers in total. A complete list of the papers in the training dataset(s) and the annotation guideline can be found in Appendix A and Appendix B, respectively.

4.4 Prototype

The prototype consists of three important components,, as illustrated in Figure 4.2. The input to the prototype is a text file, containing the first page of an academic paper, and this gets preprocessed before it is sent to the CRF and MaxEnt classifiers. If any information is extracted or classified, it is presented to the user either in console or through the ERA systems front-end, which will be introduces later in this chapter. Both classifier components contain methods for performing feature selection, training, evaluation and testing. The CRF implementation was trained on the entire dataset, while the MaxEnt implementation was only trained on 30 papers from the dataset.

CRF was chosen as the model for the IE because there has been published results where CRF has presented state-of-the-art results with IE when applied to a similar case as ours [9].

The reason MaxEnt was chosen as the text classifier was because it outperformed other classifiers during experimental testing, and for its ability to not make any independence assumption.

4.5 CRF component

CRF, presented in Section 2.4.1, is a probabilistic graphical structure which can be used for sequence labeling. CRF was used to extract nearly all fields, except the ‘Research Approach’ field. To implement the CRF model, the

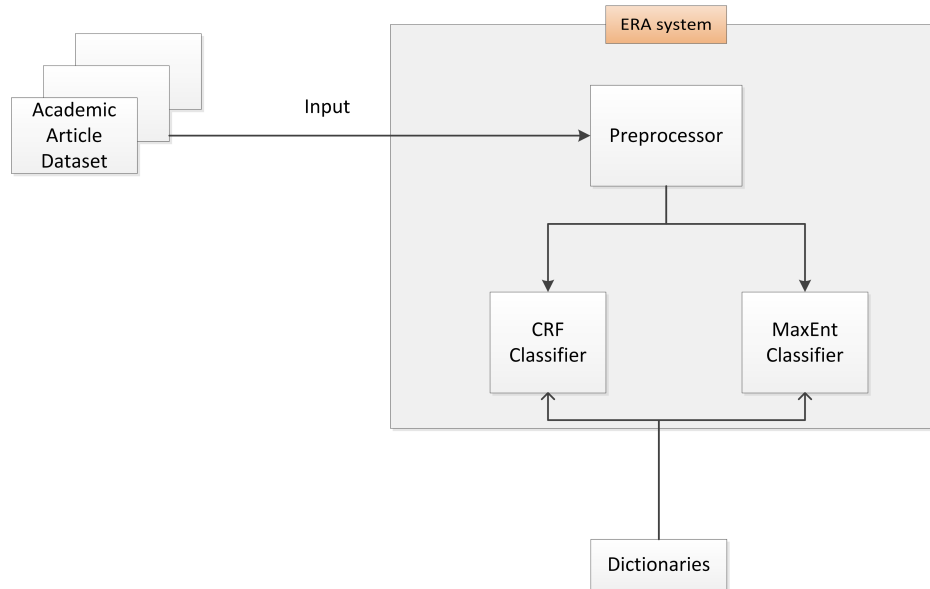


Figure 4.2: The ERA system

MALLET API¹ [32] was used, see Section 2.6.2 for an introduction, which includes methods for implementing a linear-chained CRF. The CRF still needed to be adjusted so that it could handle our purpose. This included finding suitable features to represent and distinguish the data to the CRF, also called feature engineering, as well as finding good parameters for the optimization of the machine learner.

4.5.1 Algorithm details

Several decisions had to be made regarding the CRF implementation, and with an API that provided us with the best-practice approaches, our task became more easily achievable.

To increase efficiency during training, it was important to try and get the learning method to converge as close as possible to its global maximum. To achieve this, the CRF implementation was optimized by label likelihood using L-BFGS, limited-memory Broyden-Fletcher-Goldfarb-Shanno. L-BFGS has become a popular method for non-linear optimization in CRF, because it can converge fast [16].

¹<http://mallet.cs.umass.edu/>

4.5.2 Feature Engineering

Feature engineering is a method of improving the efficiency of machine learning algorithms by reducing the dimensionality of a text or document [5]. The goal is to remove the irrelevant dimensions [33], and to do this one needs to find suitable features that can represent the important data in a distinguishable way. This is vital to the performance of the algorithm [9] and is a key aspect as to how a machine learner like CRF can make accurate predictions on new data, based on information from the features [34].

Table 4.1 shows a list of all features used in the CRF model. These features were found experimentally after several iterations.

Features	
State Transition Feature	Second Order Markov
Local Features	Feature Window Average Title Length Token First Position Token as Feature Token Char Prefix
Lexical Features	Detect Method Detect Country Detect Citation Type Detect Publisher Detect Stop Word
RegEx Features	Detect Year All Digits First Capitalized Letter

Table 4.1: CRF Features

State transition features

Deciding on the level of degree the states between the labels should be connected with each other, as illustrated in Figure 4.3, was an important decision. These are called state transition features in CRF, and they can be defined to form different Markov-orders [9]. The choice between the different

degrees of transition has a lot to say on the computational efficiency of the training. The higher Markov-orders have more transitions, and can model the dependencies better but too many transitions might cause overfitting. The lower Markov-orders on the other hand, have lesser transitions and are faster to train but might cause the training to be inefficient, due to few dependencies from neighboring labels taken into account [9].

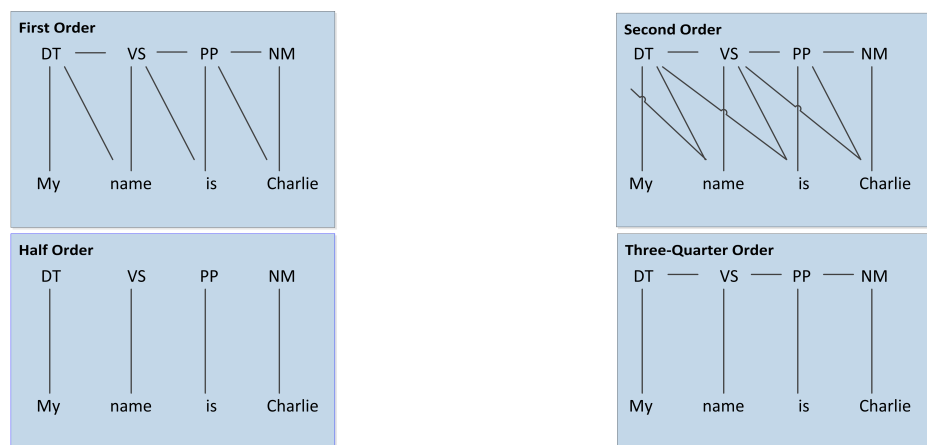


Figure 4.3: Different Degrees of State Transition

Our implementation features a second order Markov-model, with fully connected states for current and previous labels. This order was chosen because of good performance results found during experimental trials.

Local Features

The local features lets us take advantage of a variety of arbitrary information that can be found in any text. Some features are more complicated than others, and these include 'Feature Window' and 'Average Title Length'. 'Feature Window', also called sliding window, adds the features of previous and next tokens to the current token feature list. One can decide which neighboring tokens one wants to add. We used the previous token and the next two.

The 'Average Title Length' feature was added to help identify the titles of the papers. In [35], the average title length of academic software engineering papers was found to be 7.9 but we adjusted this down to 7 to try and get a higher recall value.

The rest of the features are more generic: 'Token as Feature' sets the token itself as one of the features, 'Token First Position' is assigned to the first token of a paper, 'Token Char Prefix' stores the prefix characters of each token with length 1, 2 and 3 to help identify other tokens that have the similar root but different beginning.

Lexical Features

Lexical features check a token against a list to detect whether they are a member or not. We created several lists, and these contain the names of the most typical empirical research methods, a list of all the countries and a list of well-established computer science publishers. The list with stop words was already included in the API. See Appendix B to view the lists that were created.

RegEx Features

RegEx features checks a token against a regular expression. These expressions are used to recognize patterns in a string. In our case, we used them to detect if a year was mentioned or if a token contained only digits. We also used RegEx to detect whether the token started with a capitalized letter.

Figure 4.4 illustrates an example of how a sequence of tokens can look like with features. In our case, the token sequence was converted into a feature vector sequence before using it to train the CRF model and we have several more features than in the example.

My	TOKEN_FIRST_POS	FIRST_CAP_LETTER
name		
is	STOP_WORD	
Charlie	FIRST_CAP_LETTER	

Figure 4.4: Example of a sequence of tokens and associated features

4.6 MaxEnt component

Our goal was to implement a simple, yet effective classifier that could give good results considering the small size of the training dataset. The MaxEnt classifier was used to classify what type of research approach was used in the input paper. The reason that a text classifier was used on only this field was because the other fields were mentioned specifically in the texts, and could therefore be annotated and trained on, but the research approach used by a paper was rarely mentioned explicitly in the text and would therefore have to be inferred. Our assumption was that articles using the same research approach would have some similar characteristics in the text, which could allow a text classifier to distinguish between the approaches.

The MALLET API provided us with an implementation of a MaxEnt classifier, described in Section 2.5.1. The only feature that was used was the token itself.

4.7 Preprocessing component

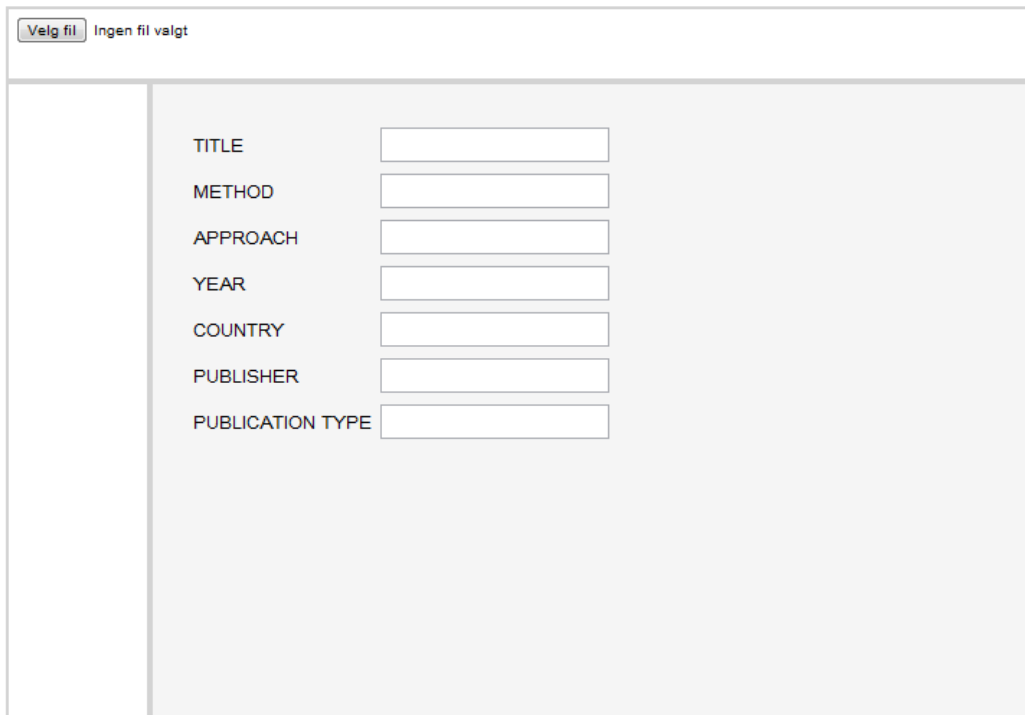
The preprocessing component handles the raw text input, and processes this into the required input-type for the CRF and MaxEnt classifier. The CRF implementation requires the data preprocessed into a sequential list of tokens and the associated labels next to them, as illustrated in Figure 4.4, except there is only one label and no features represented. The labels are processed from the text, by identifying beginning and ending XML tags. If no label is found, a default label (O) is set. The document classifier on the other hand, requires the input processed as a one instance per line. The line is in the following format:

```
[ID] [label] [text]
```

In the training phase, the [label] field was set to the extracted label from the text. Otherwise, the field was set to a default type (X). The preprocessor also handled the conversion of the text into a standard UTF-8 UNICODE encoding scheme, so that unknown characters would not be let through to the classifier components.

4.8 Implementation

The final trained models were implemented in a system to demonstrate how such a system could be used in a real case scenario. This example system was named ERA, Empirical Research Assistant. ERA serves as a proof-of-concept with regards to the problem description of this thesis, and the stated research questions. The end-result is a template form that presents the users with all the fields that were extracted, as seen in Figure 4.5.



The screenshot shows a web interface for the ERA system. At the top left, there is a button labeled "Velg fil" and the text "Ingen fil valgt". Below this, there is a vertical sidebar on the left. The main content area contains a list of labels with corresponding input fields:

TITLE	<input type="text"/>
METHOD	<input type="text"/>
APPROACH	<input type="text"/>
YEAR	<input type="text"/>
COUNTRY	<input type="text"/>
PUBLISHER	<input type="text"/>
PUBLICATION TYPE	<input type="text"/>

Figure 4.5: Screenshot of ERA

When uploading a text file, the system preprocesses the input file into the different formats needed by the two classifiers. Then, the newly created input formats are run through the trained classifiers and any fields that are extracted or classified successfully are returned. If the fields were not found, a default [N/A] label is returned.

4.8.1 Resources used

ERA was implemented in the development language Java ², using Eclipse IDE ³ as the development environment, Google Web Toolkit ⁴ for the front-end development and MALLET API ⁵ [32], see Section 2.6.2, for the classifiers.

4.8.2 System Description

Figure 4.6 and Figure 4.7 illustrate a simple view of the back-end and front-end of the ERA system.

backend.classifiers

This package contains two classes, CRF.java and MaxEnt.java, that handle the classifiers. The classes contain methods for loading the stored, and already trained models, running the input file through the feature pipes of the model, performing k-fold cross-validation, training the model and saving the model. The validation, training and saving methods can only be accessed by using the main method in the backend.main.Main.java class, and are available for future development purposes.

backend.entity

This package contains the Paper.java class, which is used to create a paper object when an input file is loaded into the system. This paper object represents the paper, and stores all the extracted information. The ID of the paper object is extracted from the filename of the input file.

backend.main

This package contains the Main.java class, which handles the initialization of the back-end system. This is where the models are loaded from the file system and the paper that is loaded gets redirected for preprocessing and

²<http://www.java.com/en/about/>

³<http://www.eclipse.org/>

⁴<http://code.google.com/intl/no-NO/webtoolkit/>

⁵<http://mallet.cs.umass.edu/>

then to the classifiers.

backend.util

This package consists of three different classes. The AvgTitlePipe.java class contains the code for the average title feature, and uses this information to create a custom pipe that is included in the CRF classifier. The IOOperations.java handles all the input/output operations needed by the system, such as writing, reading or creating text files. Finally, the PreProcessor.java class handles all the preprocessing into the different input formats.

frontend

The frond-end was created to make the testing easier, and to demonstrate that a template generating software tool for systematic reviews is an achievable goal. An in-depth description of the frond-end is not included here, but Figure 4.5 illustrates the end-result.

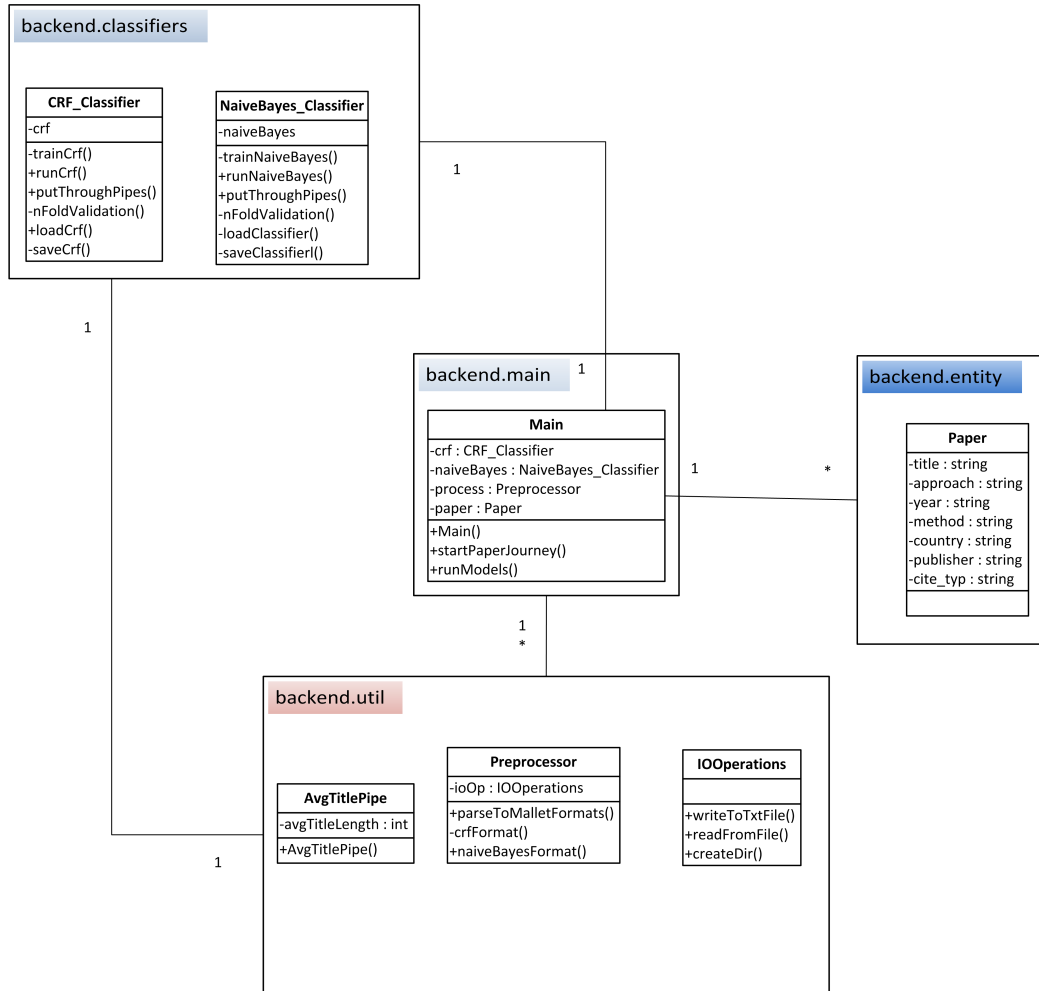


Figure 4.6: Diagram of ERA back-end system

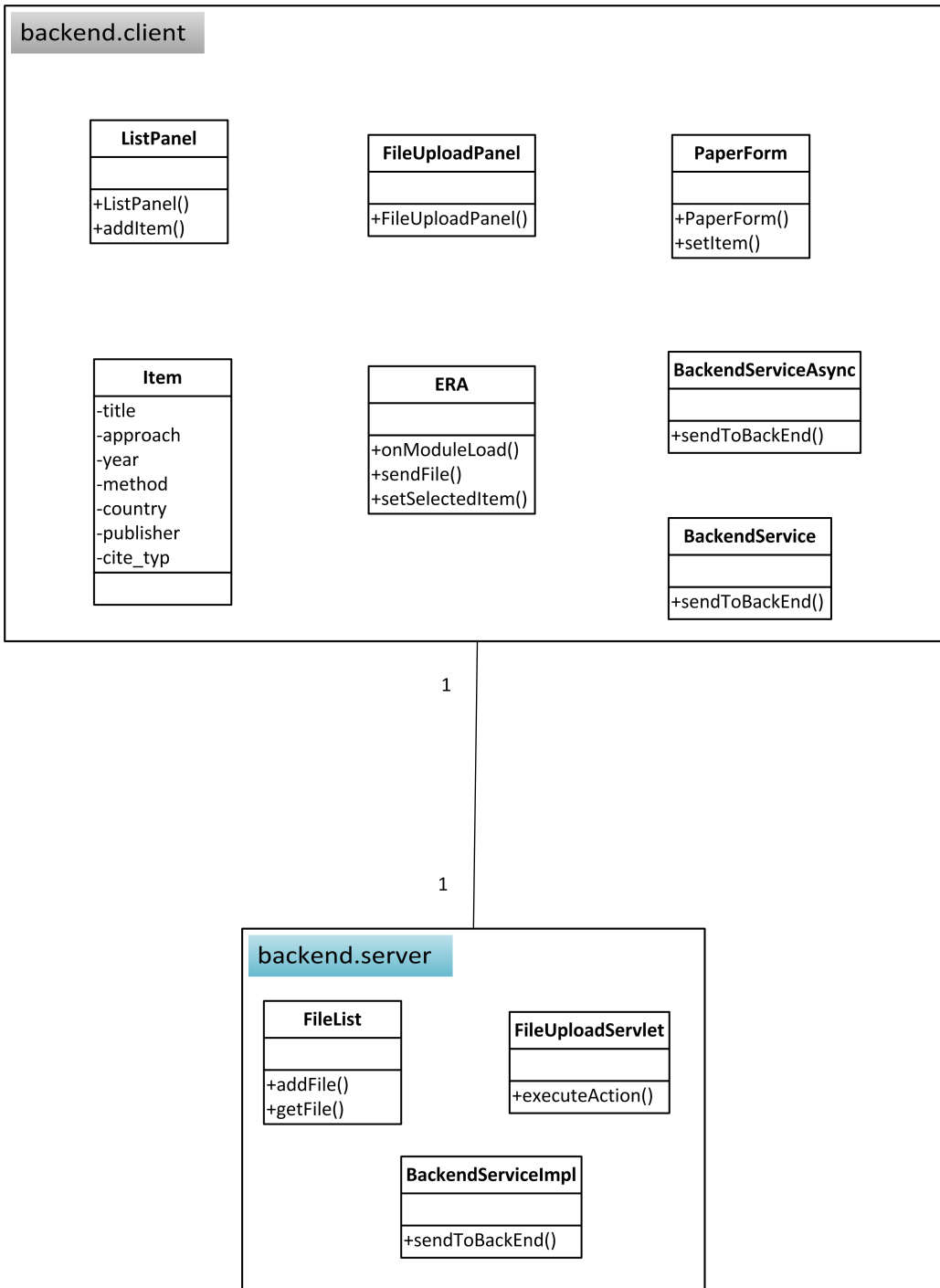


Figure 4.7: Diagram of ERA front-end system

Chapter 5

Evaluation

5.1 Introduction

This chapter will describe the evaluation methods that were used on the classifiers and present the results that were produced from the experiments. We will also argue as to why these results occurred, in the discussion.

The chapter is organized into several sections:

- Section 5.2 Explains the experiments that were conducted
- Section 5.3: Presents the experiment results
- Section 5.4 Discussion about the results

5.2 Evaluation Methods

The prototype was evaluated using standard evaluation methods for information extraction and text classification, as described in Section 2.4.2 and Section 2.5.2 respectively. Since there were no existing benchmark dataset that could be used with our proof-of-concept prototype, the dataset we created while developing the prototype was also used in our evaluations. To increase the validity of the evaluation process, both the CRF and MaxEnt implementations were trained and tested again using 5-fold cross-validation and 2-fold cross-validation, respectively. The precision, recall and F1 scores presented are the average scores across the folds. This means that the F1 scores are not calculated from the precision and recall values that are presented, but is an average score of all the F1 scores calculated in the cross-validation. The macro average scores are computed by averaging all the precision, recall or F1 values from the tables.

Table 5.1 and Table 5.2 illustrate how many times each field was identified, and annotated in the CRF and MaxEnt datasets, respectively. This gives us information about how much training data the prototype was provided with per field. The field Pub_Typ is short for publication type. All data that was used to produce the results in the evaluation experiments can be found in Appendix D.

FIELD	Annotations
TITLE	312
METHOD	26
PUBLISHER	42
YEAR	59
PUB_TYP	10
COUNTRY	62

Table 5.1: Total number of annotated tokens per field in CRF dataset

CLASS	Annotations
EMPIRICAL	7
EXPLORATORY	10
DESCRIPTIVE	13

Table 5.2: Total number of annotated tokens per field in MaxEnt dataset

5.3 Evaluation Results

This section will present the results of the evaluation experiments from the CRF and MaxEnt classifiers.

5.3.1 CRF results

The precision, recall and F1 scores for each field is presented in Table 5.3. In the table, you can see that year, publisher and country have the highest F1 scores, respectively, while method has the lowest. Figure 5.1 illustrates a graphical comparison between the different fields.

FIELD	P	R	F1
TITLE	0.7405	0.5569	0.6329
METHOD	0.9714	0.3224	0.4076
PUBLISHER	0.7489	0.7633	0.7435
YEAR	0.9270	0.7209	0.8003
PUB_TYP	1	0.5	0.5933
COUNTRY	0.9250	0.6281	0.7311
Macro Average	0.7801	0.5885	0.6702

Table 5.3: Average per field results of CRF

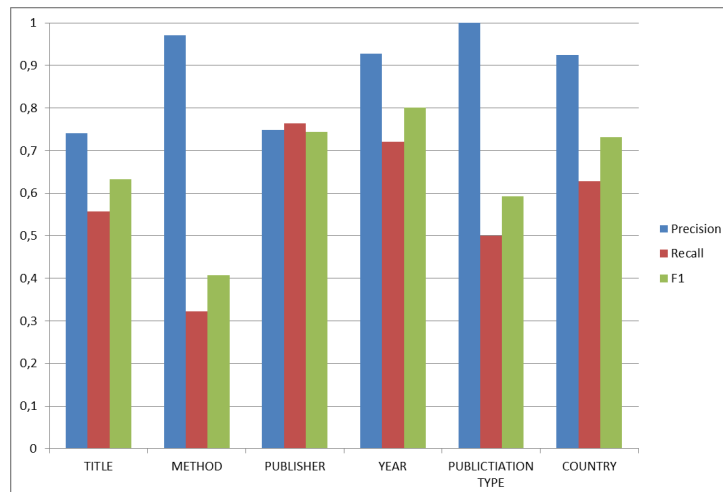


Figure 5.1: Graphical representation of field scores

In Figure 5.2, the results of testing all the different state transition degrees are presented. The results are arranged from computational easy on the left, to computational heavy on the right. The second order Markov model, see Section 4.5.2, is a computational heavier choice than the other degrees of state transition, but did not affect our implementation negatively since our dataset was not that large. But if the prototype were to use a larger dataset, it would require a more powerful system to run the training and evaluations. As can be seen in the graph, there is a significant increase in performance when selecting the second order model versus the half-order model.

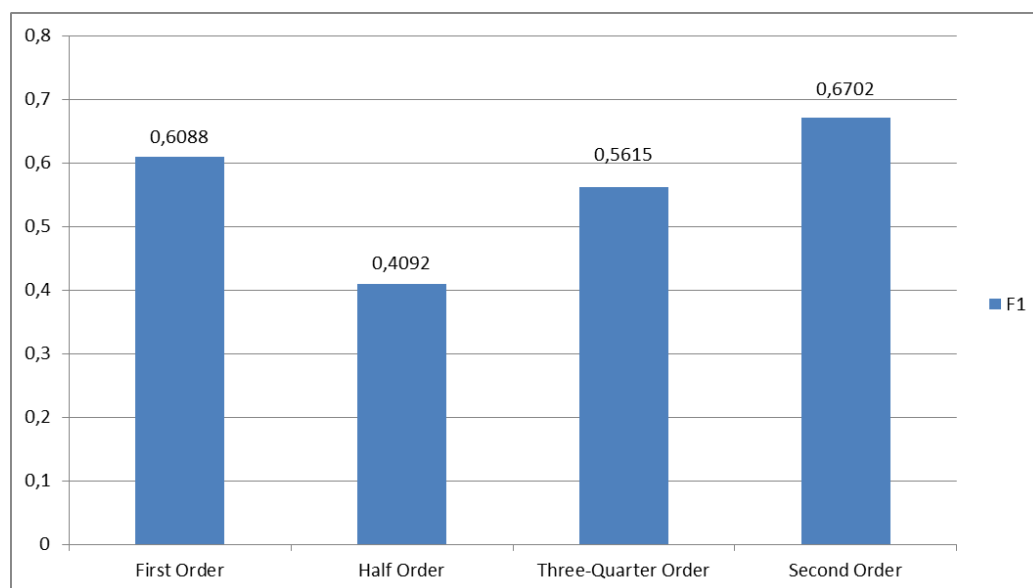


Figure 5.2: The different degrees of state transition

The graph in Figure 5.3 illustrates the effect on the macro-average precision, recall and F1 scores when omitting groups of features from the CRF classifier. Omitting the local features causes the largest drop in F1, precision and recall scores compared to when the other groups are omitted. When omitting the lexical features, there is an increase in precision, but a decrease in recall. This is the opposite of what happens when omitting the regular expression features. In this case, we can see a decrease in precision but an increase in recall.

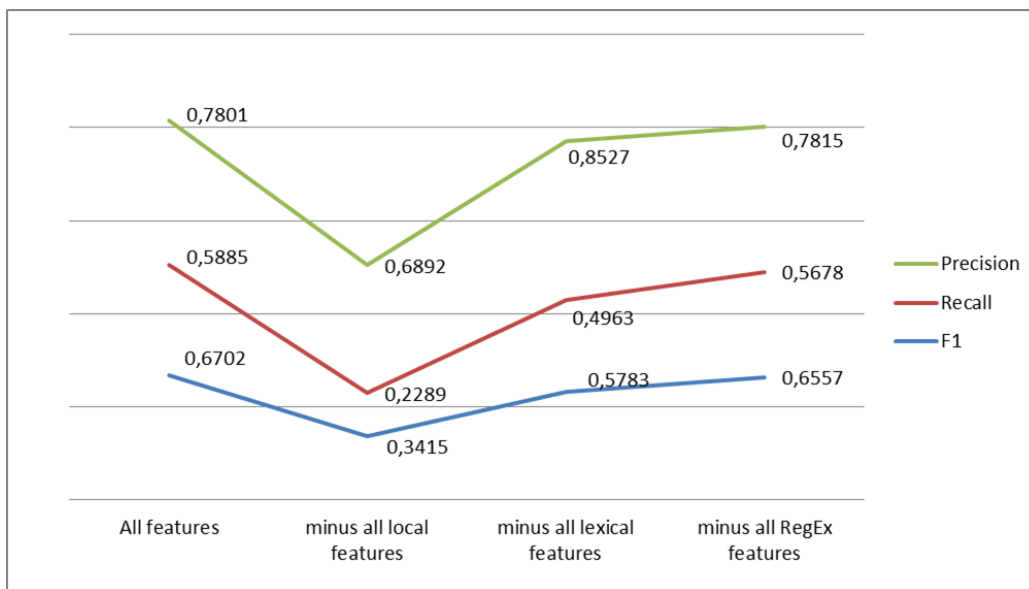


Figure 5.3: Omitting different groups of features

5.3.2 MaxEnt results

The precision, recall and F1 scores for the MaxEnt implementation is presented in Table 5.4. The 'Exploratory' class presents the highest F1 score, with 0.8167, while the 'Empirical' and 'Exploratory' classes have F1 scores of 0.6286 and 0.7692, respectively.

CLASS	P	R	F1
EMPIRICAL	0,875	0,625	0,6286
EXPLORATORY	0,75	0,9167	0,8167
DESCRIPTIVE	0,8125	0,8125	0,7692
Macro Average	0,8125	0,7847	0,7382

Table 5.4: Average per class results of MaxEnt

The reason MaxEnt was chosen as the classifier is presented in Figure 5.4. This comparison of the three different text classifiers show that MaxEnt delivers the highest scores, compared to the two other classifiers. NB was close to MaxEnt, with MaxEnt only performing 0.0096 (1.32% increase) better, while MaxEnt outperformed DT with 0.1912 (35% increase) difference in

score.

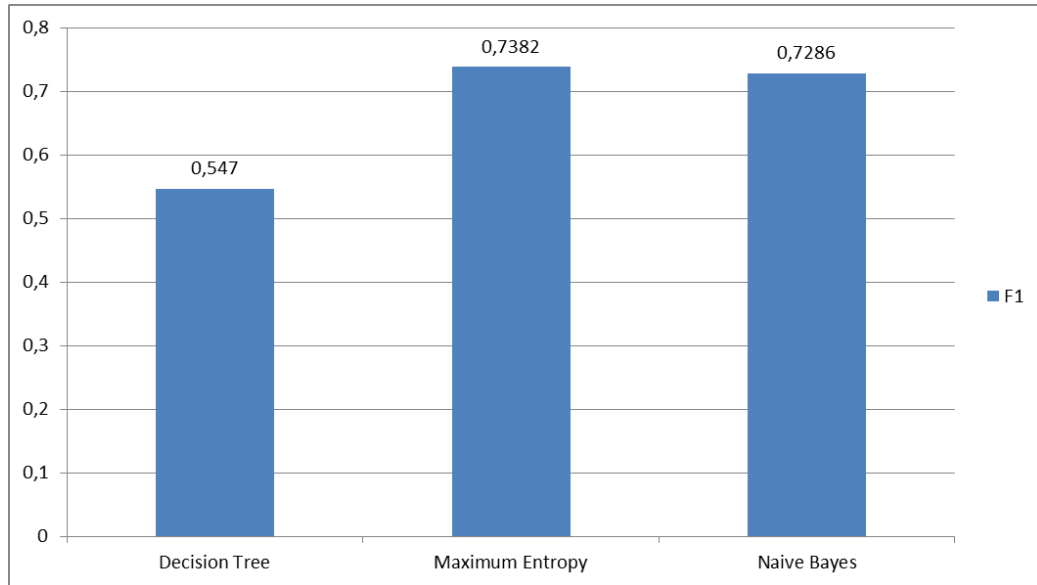


Figure 5.4: Comparison of different classifiers

Table 5.5 presents the confusion matrix of the MaxEnt. Seven occurrences of 'Empirical' were predicted correctly, and one occurrence was misclassified as 'Descriptive'. The class 'Exploratory' was successfully classified ten times, and misclassified twice as 'Descriptive'. All ten of the 'Descriptive' occurrences were predicted correctly, with the addition of two more occurrences misclassified into this class.

		Actual			Total
		Empirical	Exploratory	Descriptive	
Predicted	Empirical	7	.	.	7
	Exploratory	.	10	.	10
	Descriptive	1	2	10	13
Total		8	12	10	

Table 5.5: MaxEnt confusion matrix

5.4 Discussion

With a macro-average F1 score of 0.6702, the CRF implementation has proven itself to function good at its tasks, yet still has potential for improvement. Much of this potential for improvement is within the strong connection the scores have with the number of annotated tokens the CRF had available to use during training, as well as the features used to distinguish these different fields.

The 'year' field produced the best scores, and had the most positive effect on the overall F1 score, with an average F1 score of 0.8003. The 'method' field did not have a positive effect on the overall F1 scores, but in fact had a negative impact, with an average F1 score of 0.4076. This is mostly because the field only had 26 annotated tokens, over half as much as the best performing field. All the field that had F1 scores above 0.6, had over 42 or more annotated tokens while the two lowest performing fields only had from 26 and below. One could argue that the CRF implementation would benefit from a larger dataset, that had a larger representation of the 'method' and 'pub_typ' fields. Of course, this would also mean more noise data that would be included during training. Still, considering the small training data, the extraction of the fields produced good results. Papers that contain most, if not all of the fields would be the perfect training data but will not always be present in a dataset.

While the local features demonstrated the large impact they had on the overall scores, as illustrated in Figure 5.2, the lexical and RegEx features only demonstrated small improvements to the scores. There was even an increase in precision score when omitting all RegEx features. The reason for this is that many of the local features are designed to boost specific fields, like the average title length feature, which in the end helps to increase the overall scores. One could argue that the CRF implementation could omit the RegEx features, and instead try to engineer field-specific features instead. The results clearly show that our use of these local features are one of the key aspects to why the CRF performed as it did.

The importance of choosing the degree of state transitions is evident in Figure 4.3. Since there were a low number of labeled tokens, it was even more important to consider the previous and next labels. This is the reason why the second order Markov model performed with the highest results.

The MaxEnt implementation delivered a macro-average F1 score of 0.7382.

This demonstrates how a simple text classifier can produce good results, considering the small size of the dataset, the number of labels available for training and that only one feature was used. Nonetheless, this implementation does need improvement, as a total of three papers were misclassified and the recall scores for the class 'Empirical' were low. We believe the small amount of labeled dataset that the class 'Empirical' had available has affected the score. Since the CRF implementation demonstrated the importance of feature engineering, we believe that the MaxEnt implementation could benefit from experimentation with more features.

Concerning the research questions, we argue that both the results presented and the implementation of the classifiers into the ERA system show that it is possible. While McCullum and Feng [9] presented results showing the use of CRF with IE, we have implemented a system using two trained classifiers, on a small self-annotated dataset. Our research has proven it to be practically achievable. However, there are still limitations to our research if used in a real scenario. The training data is too small for it to be representative of all software engineering papers, the specific information each SLR wants to retrieve differs and only one person annotated the dataset. Still, as a proof-of-concept we argue that we have demonstrated it to be achievable.

Chapter 6

Conclusion & Future Work

6.1 Introduction

This chapter will present the conclusion of the thesis, answer the research questions and give suggestions to possible future work which can be done regarding our research and prototype.

The chapter is organized into two sections:

- Section 6.2 Presents the conclusion of the thesis
- Section 6.3: Suggestions about possible future work

6.2 Conclusion

The goal of this thesis was to investigate if one could develop a method for automatic literature reviews. Relevant theory into the subjects of SLR, IE and TC was researched and state-of-the-art techniques were found. Existing research and solutions were analyzed, but found to be insufficient at solving our problem. A prototype was developed as a proof-of-concept, and tested against a dataset that we created, and the results were evaluated and finally discussed.

The contribution of this thesis is the researched, developed and evaluated prototype named ERA. The Empirical Research Assistant was developed over the course of several iterations, each iteration aiming to expand its labeled dataset and engineer features to achieve better results. The labeled dataset was annotated for the purpose of this thesis, and did not exist beforehand. CRF and MaxEnt were used for the extraction and classification, each machine learner assigned to infer different fields. These fields were found to be of use to researchers when conducting an SLR, by examining two concluded SLRs [30, 4]. The prototype presented good results during the evaluations.

With the research questions in mind, we have provided direct answers to each question:

Main Question: *Could one develop a method for automatic systematic literature reviews?*

A: Our research has shown that a crucial part of an SLR can be automated, but there is still much work that needs to be done before an entire SLR can be conducted automatically.

RQ1: *Is it possible to develop a system/tool that could support the data extraction process of a systematic literature review?*

A: The ERA system has demonstrated that it is possible, with the use of IE and TC techniques and trained machine learning models.

RQ2: *Are there any existing solutions and/or systems that solve the same problems addressed in this thesis?*

A: Although there are several similar systems, they are not compatible at solving our task of assisting in SLRs.

RQ3: *How well does the proposed solution work?*

A: Our proposed solutions IE component has an average F1 performance score of 67.02%, while our TC component delivers an average F1 performance score of 73.82%.

Our conclusion is that it is possible to develop a method to support the data extraction step of an SLR, as supported by this thesis and the prototype. The prototype does not need to be trained on a large dataset to produce good results, as our evaluations have demonstrated.

6.3 Future Work

With the discussion and limitations of the prototype in mind, we have suggested some future work:

- The ERA systems dataset could be extended. A larger dataset would allow the machine learners to be trained on more cases where the fields appear, and this might in return improve their ability to detect these fields.
- Testing more methods and new state-of-the-art methods could prove to be beneficial.
- Conducting more specific feature engineering, directed at the fields that were more difficult to detect. Especially the MaxEnt component could benefit from experimenting with different features.
- A case study, using the ERA system as a support tool for the data extraction step could provide valuable insight into how the the system would work in a real-case scenario, and how it could be improved further.

Appendix A

Dataset

A.1 CRF Dataset

ID	SLR	Title
S3	[30]	Challenges in requirements engineering for mobile games development: the meantime case study
S5	[30]	Evaluation of object-oriented design patterns in game development
S11	[30]	Requirements engineering and the creative process in the video game industry
S12	[30]	Emotional requirements in video games
S13	[30]	A practical implementation of a 3-D game engine
S18	[30]	The usability of massively multiplayer online role-playing games: designing for new users
S19	[30]	ScriptEase: a generative/adaptive programming paradigm for game scripting
S24	[30]	Component based game development - a solution to escalating costs and expanding deadlines
S25	[30]	Ucigame, a java library for games
S28	[30]	PlayMancer - a serious gaming 3D environment
S31	[30]	Establishing user requirements: incorporating gamer preferences into interactive games design
S32	[30]	Pervasive game flow: understanding player enjoyment in pervasive gaming
S33	[30]	The platform of quick development of mobile 3D game
S35	[30]	Design and implementation of a multiplayer bluetooth game
S37	[30]	Playability heuristics for mobile multi-player games
S38	[30]	Playability heuristics for mobile games
S51	[30]	Using prototypes in early pervasive game development
S53	[30]	Life-cycle of the games industry - the specificities of creative industries
S58	[30]	Using genres to customize usability evaluations of video games
S66	[30]	From usability to playability - introduction to player-centred video game development process
S69	[30]	User experience in interactive computer game development
S72	[30]	Computer game - flow design
S74	[30]	Creating an emotionally adaptive game
S78	[30]	Guidelines for designing augmented reality games

S81	[30]	Empirical validation of test-driven pair programming in game development
S82	[30]	Capturing player enjoyment in computer games
S83	[30]	Mobile game development: object orientation or not
S84	[30]	Object-orientation is evil to mobile game - experience from industrial mobile RPGs
ID2	[4]	Quantitative studies in software release planning under risk and resource constraints
ID3	[4]	Trade-off analysis for requirements selection
ID4	[4]	An analytical model for requirements selection quality evaluation in product software development
ID5	[4]	Software release planning: an evolutionary and iterative approach
ID7	[4]	Intelligent support for software release planning
ID8	[4]	Release planning under fuzzy effort constraints
ID9	[4]	Supporting software release planning decisions for evolving systems
ID10	[4]	Determination of the next release of a software product: an approach using integer linear programming
ID11	[4]	Fuzzy structural dependency constraints in software release planning
ID12	[4]	Measuring dependency constraint satisfaction in software release planning using dissimilarity of fuzzy graphs

Table A.1: All the papers used as CRF dataset

A.2 MaxEnt Dataset

ID	SLR	Title
S03	[30]	Challenges in requirements engineering for mobile games development: the meantime case study
S11	[30]	Requirements engineering and the creative process in the video game industry
S12	[30]	Emotional requirements in video games
S18	[30]	The usability of massively multiplayer online role-playing games: designing for new users
S24	[30]	Component based game development - a solution to escalating costs and expanding deadlines

S31	[30]	Establishing user requirements: incorporating gamer preferences into interactive games design
S33	[30]	The platform of quick development of mobile 3D game
S35	[30]	Design and implementation of a multiplayer bluetooth game
S37	[30]	Playability heuristics for mobile multi-player games
S38	[30]	Playability heuristics for mobile games
S51	[30]	Using prototypes in early pervasive game development
S58	[30]	Using genres to customize usability evaluations of video games
S66	[30]	From usability to playability - introduction to player-centred video game development process
S69	[30]	User experience in interactive computer game development
S74	[30]	Creating an emotionally adaptive game
S78	[30]	Guidelines for designing augmented reality games
S81	[30]	Empirical validation of test-driven pair programming in game development
S82	[30]	Capturing player enjoyment in computer games
S83	[30]	Mobile game development: object orientation or not
S84	[30]	Object-orientation is evil to mobile game - experience from industrial mobile RPGs
ID02	[4]	Quantitative studies in software release planning under risk and resource constraints
ID03	[4]	Trade-off analysis for requirements selection
ID4	[4]	An analytical model for requirements selection quality evaluation in product software development
ID5	[4]	Software release planning: an evolutionary and iterative approach
ID7	[4]	Intelligent support for software release planning
ID8	[4]	Release planning under fuzzy effort constraints
ID9	[4]	Supporting software release planning decisions for evolving systems
ID10	[4]	Determination of the next release of a software product: an approach using integer linear programming
ID11	[4]	Fuzzy structural dependency constraints in software release planning
ID12	[4]	Measuring dependency constraint satisfaction in software release planning using dissimilarity of fuzzy graphs

Table A.2: All the papers used as MaxEnt dataset

Appendix B

Annotation Guideline

Read through this document once before starting the annotation process.

Intro

Before you continue with the steps below you should open the zipped file named "Labeling_Task" and extract its content.

The zip file contains a folder named "Dataset", a text document named "Annotation_Guidelines" and an Excel form named "Text_Classification". Extract this folder onto your desktop for easy access.

"Dataset" folder contains several documents that you will be annotating. Each of these documents contains the first page of a research paper in raw text. Use a common text editor, like Notepad, to open these documents.

The Excel form named "Text_Classification" is to be used when annotating the field "Approach". The Excel form has an ID field, which is the document name itself, like "S2-4" or "S2-68". The "Approach" field in the Excel document is where the label is to be assigned.

Annotations

There are 7 fields that, if identified in the text, must be annotated with XML tags. There is 1 field that requires you to write down the annotation in the Excel form, without XML tags.

The text must not be altered in any way to better match a description of a label. Only annotate the fields that you identify and match with the definitions below:

XML Tags

Title:

Annotate the entire title of the paper, including any subtitles. E.g. “The use of nuclear-powered servers: Next generation server-farms?”

Label: <TITLE> ... </TITLE>

Research Method:

Annotate if these three empirical research methods are explicitly mentioned in the text as used in the paper; “case study”, “experiment” or “survey”.

Label: <METHOD> ... </METHOD>

Country:

Annotate the papers’ country of origin if it occurs, e.g. ”Spain”, or ”Singapore”. Only annotate the country of origin in the appropriate context, like when it appears after the papers author(s) or institute(s)/affiliation(s). If multiple country of origin exists, annotate them as well.

Label: <COUNTRY> ... </COUNTRY>

Publisher:

Annotate the publisher of the paper, e.g. ”IEEE” or ”ACM” or “Elsevier”. Only annotate the “Elsevier” part if this publisher occurs.

Label: <PUBLISHER> ... </PUBLISHER>

Year:

Annotate the year of publication e.g. ”2008”. Annotate all occurrences of this field in the proper context, e.g. when mentioned next to publisher or author.

Label: <YEAR> ... </YEAR>

Type of Publication:

Annotate what type of publication the paper is, only if explicitly mention as either “journal”, “workshop” or “conference”

Label: <CIT_TYP> ... </ CIT_TYP >

Excel form

By just reading the *abstract* part of the document, derive from the definitions below what type of research approach the document uses:

Empirical:

Whether the findings in the research are from direct or indirect observations, such as a case study [30] .

Descriptive:

If a system, tool or method is presented and described in the research [30] .

Exploratory:

If the research investigates a problem that has not been defined, and helps to determine the best research way or design, the best data collection method and the best method for selecting subjects. [30]

Appendix C

Lexicon Lists

C.1 Countries

The list was retrieved, and altered from: http://openconcept.ca/blog/mgifford/text_list_all_countries

Afghanistan Albania Algeria Andorra Angola Antigua Argentina Armenia Australia Austria Azerbaijan Bahamas Bahrain Bangladesh Barbados Belarus Belgium Belize Benin Bhutan Bolivia Bosnia Botswana Brazil Brunei Bulgaria Burkina Burundi Cambodia Cameroon Canada Cape Verde Central African Republic Chad Chile China Colombia Comoros Congo Congo Costa Rica Croatia Cuba Cyprus Czech Republic Denmark Djibouti Dominica Dominican Republic East Timor Ecuador Egypt El Salvador Equatorial Guinea Eritrea Estonia Ethiopia Fiji Finland France Gabon Gambia Georgia Germany Ghana Greece Grenada Guatemala Guinea Guinea Bissau Guyana Haiti Honduras Hungary Iceland India Indonesia Iran Iraq Ireland Israel Italy Ivory Coast Jamaica Japan Jordan Kazakhstan Kenya Kiribati Korea North Korea South Kosovo Kuwait Kyrgyzstan Laos Latvia Lebanon Lesotho Liberia Libya Liechtenstein Lithuania Luxembourg Macedonia Madagascar Malawi Malaysia Maldives Mali Malta Marshall Islands Mauritania Mauritius Mexico Micronesia Moldova Monaco Mongolia Montenegro Morocco Mozambique Myanmar Namibia Nauru Nepal Netherlands New Zealand Nicaragua Niger Nigeria Norway Oman Pakistan Palau Panama Papua New Guinea Paraguay Peru Philippines Poland Portugal Qatar Romania Russia Rwanda St Kitts Nevis St Lucia Saint Vincent the Grenadines

Samoa San Marino Sao Tome Principe Saudi Arabia Senegal Serbia Seychelles Sierra Leone Singapore Slovakia Slovenia Solomon Islands Somalia South Africa Spain Sri Lanka Sudan Suriname Swaziland Sweden Switzerland Syria Taiwan Tajikistan Tanzania Thailand Togo Tonga Trinidad Tunisia Turkey Turkmenistan Tuvalu Uganda Ukraine Emirates United Kingdom UK United States USA US Uruguay Uzbekistan Vanuatu Vatican Venezuela Vietnam Yemen Zambia Zimbabwe

C.2 Research Methods

case study studies experiment experiments survey surveys

C.3 Publication Types

conference journal workshop

C.4 Publishers

ieee ACM springerverlag springer elsevier

Appendix D

Evaluation Data

This data has been re-sized and presented in a reader-friendly way, but NOT altered for the appendix. All raw data from the experiments presented in the Evaluation chapter will be attached to the thesis in a zip file, inside a folder named 'Validation Data'. We recommend searching for the string "CRF Trained with Convergence!" to find the evaluation data in the raw texts, as there is a lot of calculations and data in the texts.

D.1 CRF data

Iteration 1:

test tokenaccuracy=0,9912

TITLE

test segments true=80 pred=66 correct=32 misses=48 alarms=34

test precision=0,4848 recall=0,4 f1=0,4384

METHOD

test segments true=6 pred=0 correct=0 misses=6 alarms=0

test precision=1 recall=0 f1=0

PUBLISHER

test segments true=10 pred=6 correct=4 misses=6 alarms=2

test precision=0,6667 recall=0,4 f1=0,5

YEAR

test segments true=12 pred=10 correct=9 misses=3 alarms=1

test precision=0,9 recall=0,75 f1=0,8182

CIT_TYP

test segments true=1 pred=0 correct=0 misses=1 alarms=0

test precision=1 recall=0 f1=0

COUNTRY

test segments true=11 pred=4 correct=3 misses=8 alarms=1

test precision=0,75 recall=0,2727 f1=0,4

OVERALL

test segments true=120 pred=86 correct=48 misses=72 alarms=38

test precision=0,5581 recall=0,4 f1=0,466

Iteration 3:

test tokenaccuracy=0,9952

TITLE

test segments true=58 pred=50 correct=50 misses=8 alarms=0

test precision=1 recall=0,8621 f1=0,9259

METHOD

test segments true=11 pred=7 correct=6 misses=5 alarms=1

test precision=0,8571 recall=0,5455 f1=0,6667

PUBLISHER

test segments true=5 pred=6 correct=5 misses=0 alarms=1

test precision=0,8333 recall=1 f1=0,9091

YEAR

test segments true=6 pred=5 correct=5 misses=1 alarms=0

test precision=1 recall=0,8333 f1=0,9091

CIT_TYP

test segments true=3 pred=1 correct=1 misses=2 alarms=0

test precision=1 recall=0,3333 f1=0,5

COUNTRY

test segments true=8 pred=8 correct=7 misses=1 alarms=1

test precision=0,875 recall=0,875 f1=0,875

OVERALL

test segments true=91 pred=77 correct=74 misses=17 alarms=3

test precision=0,961 recall=0,8132 f1=0,881

Iteration 3:

test tokenaccuracy=0,987

TITLE

test segments true=68 pred=53 correct=26 misses=42 alarms=27
test precision=0,4906 recall=0,3824 f1=0,4298

METHOD

test segments true=3 pred=2 correct=2 misses=1 alarms=0
test precision=1 recall=0,6667 f1=0,8

PUBLISHER

test segments true=12 pred=10 correct=8 misses=4 alarms=2
test precision=0,8 recall=0,6667 f1=0,7273

YEAR

test segments true=14 pred=8 correct=8 misses=6 alarms=0
test precision=1 recall=0,5714 f1=0,7273

CIT_TYP

test segments true=3 pred=2 correct=2 misses=1 alarms=0
test precision=1 recall=0,6667 f1=0,8

COUNTRY

test segments true=11 pred=6 correct=6 misses=5 alarms=0
test precision=1 recall=0,5455 f1=0,7059

OVERALL

test segments true=111 pred=81 correct=52 misses=59 alarms=29
test precision=0,642 recall=0,4685 f1=0,5417

Iteration 4:

test tokenaccuracy=0,9929

TITLE

test segments true=45 pred=33 correct=24 misses=21 alarms=9
test precision=0,7273 recall=0,5333 f1=0,6154

METHOD

test segments true=1 pred=0 correct=0 misses=1 alarms=0
test precision=1 recall=0 f1=0

PUBLISHER

test segments true=8 pred=9 correct=6 misses=2 alarms=3
test precision=0,6667 recall=0,75 f1=0,7059

YEAR

test segments true=15 pred=9 correct=8 misses=7 alarms=1
test precision=0,8889 recall=0,5333 f1=0,6667

CIT_TYP

test segments true=1 pred=1 correct=1 misses=0 alarms=0
test precision=1 recall=1 f1=1
COUNTRY
test segments true=17 pred=11 correct=11 misses=6 alarms=0
test precision=1 recall=0,6471 f1=0,7857
OVERALL
test segments true=87 pred=63 correct=50 misses=37 alarms=13
test precision=0,7937 recall=0,5747 f1=0,6667

Iteration 5:

test tokenaccuracy=0,9932
TITLE
test segments true=61 pred=37 correct=37 misses=24 alarms=0
test precision=1 recall=0,6066 f1=0,7551
METHOD
test segments true=5 pred=2 correct=2 misses=3 alarms=0
test precision=1 recall=0,4 f1=0,5714
PUBLISHER
test segments true=7 pred=9 correct=7 misses=0 alarms=2
test precision=0,7778 recall=1 f1=0,875
YEAR
test segments true=12 pred=13 correct=11 misses=1 alarms=2
test precision=0,8462 recall=0,9167 f1=0,88
CIT_TYP
test segments true=2 pred=1 correct=1 misses=1 alarms=0
test precision=1 recall=0,5 f1=0,6667
COUNTRY
test segments true=15 pred=12 correct=12 misses=3 alarms=0
test precision=1 recall=0,8 f1=0,8889
OVERALL
test segments true=102 pred=74 correct=70 misses=32 alarms=4
test precision=0,9459 recall=0,6863 f1=0,7955

D.2 MaxEnt data

Iteration 1:

Class 'EMPIRICAL': P: 1.0 R: 0.25 F1: 0.4

Class 'EXPLORATORY': P: 0.8333333333333334 R: 0.8333333333333334 F1: 0.8333333333333334

Class 'DESCRIPTIVE': P: 0.625 R: 1.0 F1: 0.7692307692307693

Confusion Matrix, row=true, column=predicted accuracy=0.7333333333333333

		Actual			Total
		Empirical	Exploratory	Descriptive	
Predicted	Empirical	1	1	2	4
	Exploratory	.	5	1	6
	Descriptive	.	.	5	5
Total		1	6	8	

Table D.1: Iteration 1 Confusion Matrix

Iteration 2:

Class 'EMPIRICAL': P: 0.75 R: 1.0 F1: 0.8571428571428571

Class 'EXPLORATORY': P: 0.6666666666666666 R: 1.0 F1: 0.8

Class 'DESCRIPTIVE': P: 1.0 R: 0.625 F1: 0.7692307692307693

Confusion Matrix, row=true, column=predicted accuracy=0.8

		Actual			Total
		Empirical	Exploratory	Descriptive	
Predicted	Empirical	3	.	.	3
	Exploratory	.	4	.	4
	Descriptive	1	2	5	8
Total		4	6	5	

Table D.2: Iteration 2 Confusion Matrix

Trial:

		Actual			Total
		Empirical	Exploratory	Descriptive	
Predicted	Empirical	7	.	.	7
	Exploratory	.	10	.	10
	Descriptive	1	2	10	13
Total		8	12	10	

Table D.3: Trial Confusion Matrix

Accuracy: 0.9
Precision EMPIRICAL': 0.875
Recall: EMPIRICAL': 1.0
Precision EXPLORATORY': 0.8333333333333334
Recall: EXPLORATORY': 1.0
Precision DESCRIPTIVE': 1.0
Recall: DESCRIPTIVE': 0.7692307692307693

Confusion Matrix, row=true, column=predicted accuracy=0.9

Bibliography

- [1] B. Kitchenham, “Guidelines for performing systematic literature reviews in software engineering,” Keele University and University of Durham, ESBE Technical Report 2.3, July 2007.
- [2] A. White and K. Schmidt, “Systematic literature reviews,” *Complementary Therapies in Medicine*, vol. 13, no. 1, pp. 54–60, 2005.
- [3] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, 2007.
- [4] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, and M. U. Shafique, “A systematic review on strategic release planning models,” *Information and Software Technology*, vol. 52, pp. 237–248, March 2010.
- [5] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [6] S. Sarawagi, “Information extraction,” *Foundations and Trends in Databases*, vol. 1, pp. 261–377, 2008.
- [7] K. Kaiser and S. Miksch, “Information extraction: A survey,” Vienna University of Technology, Tech. Rep., May 2005.
- [8] D. Cruzes, V. Basili, F. Shull, and M. Jino, “Automated information extraction from empirical software engineering literature: Is that possible?” in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium On*, 2007, pp. 491–493.

- [9] F. Peng and A. McCallum, “Accurate information extraction from research papers using conditional random fields,” in *HLT-NAACL04*, 2004, pp. 329–336.
- [10] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [11] D. Freitag and A. K. McCallum, “Information extraction with hmms and shrinkage,” in *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999, pp. 31–36.
- [12] J. Su, G. Zhou, and G. Zhou, “Named entity recognition using an hmm-based chunk tagger,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL ’02. Association for Computational Linguistics, 2002, pp. 473–480.
- [13] A. McCallum and D. Freitag, “Maximum entropy markov models for information extraction and segmentation,” in *Proceedings of the 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 591–598.
- [14] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML ’01. Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.
- [15] B. Settles, “ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text,” *Bioinformatics*, vol. 21, no. 14, pp. 3191–3192, 2005.
- [16] F. Sha and F. Pereira, “Shallow parsing with conditional random fields,” in *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, 2003, pp. 213–220.
- [17] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *International Joint Conference ON artificial intelligence*, 1995, pp. 1137–1143.

-
- [18] D. T. Larose, *Discovering Knowledge in Data, an Introduction to Data Mining*. Wiley-Interscience, 2005.
- [19] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.
- [20] L. Hyafil and R. L. Rivest, “Constructing optimal binary decision trees is np-complete,” *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.
- [21] D. D. Lewis, “Naive (bayes) at forty: The independence assumption in information retrieval,” in *Machine Learning: ECML-98*. Springer Verlag, 1998, pp. 4–15.
- [22] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, “Some effective techniques for naive bayes text classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 11, pp. 1457–1466, 2006.
- [23] K. Nigam, J. Lafferty, and A. McCallum, “Using maximum entropy for text classification,” 1999.
- [24] H. H. C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox, “Automatic document metadata extraction using support vector machines,” in *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, 2003, pp. 37–48.
- [25] Y. Hu, “Automatic extraction of titles from general documents using machine learning,” in *Proceedings of ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, 2005, pp. 145–154.
- [26] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano, “Text mining for product attribute extraction,” *SIGKDD Explorations*, vol. 1, pp. 41–48, 2006.
- [27] X. Lu, B. Kahle, J. Z. Wang, and C. L. Giles, “A metadata generation system for scanned scientific volumes,” in *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*. ACM, 2008, pp. 167–176.

- [28] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: an automatic citation indexing system," in *International Conference On Digital Libraries*, 1998, pp. 89–98.
- [29] H.-M. Müller, E. E. Kenny, and P. W. Sternberg, "Textpresso: An ontology-based information retrieval and extraction system for biological literature," *PLoS Biol*, vol. 2, p. 309, 2004.
- [30] A. Ampatzoglou and I. Stamelos, "Software engineering research for computer games: A systematic review," *Information and Software Technology*, vol. 52, pp. 888–901, 2010.
- [31] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Software*, vol. 12, no. 4, pp. 52–62, 1995.
- [32] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002, <http://mallet.cs.umass.edu>.
- [33] C. Ding, X. He, H. Zha, and H. Simon, "Adaptive dimension reduction for clustering high dimensional data," in *Proceedings. 2002 IEEE International Conference on Data Mining, 2002. ICDM 2003.*, 2002, pp. 147–154.
- [34] T. G. Dietterich, "Machine learning for sequential data: A review6," in *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, 2002, pp. 15–30.
- [35] L. Anthony, "Characteristic features of research article titles in computer science," *Professional Communication, IEEE Transactions on*, vol. 44, no. 3, pp. 187–194, 2001.