Catching Criminals by Chance

A Probabilistic Approach to Named Entity Recognition using Targeted Feedback

Jasper Kuperus
jasper@jasperkuperus.nl

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
University of Twente, Enschede, The Netherlands

Thesis submitted to obtain the degree of
Master of Science in Computer Science
with specialization
Information Systems Engineering

Date:
June 15, 2012

University of Twente, EEMCS
Enschede, The Netherlands
Dr. ir. Maurice van Keulen
Dr. ir. Dolf Trieschnigg
Mena B. Habib MSc

Netherlands Forensic Institute, Kecida
The Hague, The Netherlands
Dr. Cor J. Veenman

Acknowledgements

I would like to thank a few people for helping and supporting me during the course of this research project.

First of all, I would like to thank my supervisors: Maurice, Cor, Dolf and Mena. Thank you for the feedback and input and keeping me motivated for the subject. The meetings were very useful and often ended in extensive brainstorm sessions. Maurice, in particular, I would like to thank for introducing me to the NFI, his constant enthusiasm and patience.

I would also like to thank my colleagues at the NFI, Kecida. I had a great time and it was a real pleasure to execute my research project in such a nice and interesting team. In particular, I would like to thank Maarten and Ankie for introducing me to the subject of NER and demonstrating their tool. Also, I would like to thank Ranieri for replacing the frustrations of public transport with carpooling and interesting discussions.

Then, I would like to thank the people of SoNaR for providing me a pre-release of the 1 million words corpus for my research project. In particular, I would like to thank Franciska de Jong, Thijs Verschoor, Martin Reynaert and Bart Desmet. Being able to use this corpus in this research project saved me the time of creating my own corpus to validate my approach on.

Finally, I would like to thank my friends, family and girlfriend for supporting me. In particular I would like to thank Jorrit and Thomas for thinking along with me when encountering specific problems.

Jasper Kuperus
June 15, 2012

Abstract

In forensics, large amounts of unstructured data have to be analyzed in order to find evidence or to detect risks. For example, the contents of a personal computer or USB data carriers belonging to a suspect. Automatic processing of these large amounts of unstructured data, using techniques like Information Extraction, is inevitable. Named Entity Recognition (NER) is an important first step in Information Extraction and still a difficult task.

A main challenge in NER is the ambiguity among the extracted named entities. Most approaches take a hard decision on which named entities belong to which class or which boundary fits an entity. However, often there is a significant amount of ambiguity when making this choice, resulting in errors by making these hard decisions. Instead of making such a choice, all possible alternatives can be preserved with a corresponding confidence of the probability that it is the correct choice. Extracting and handling entities in such a probabilistic way is called *Probabilistic Named Entity Recognition* (PNER).

Combining the fields of Probabilistic Databases and Information Extraction results in a new field of research. This research project explores the problem of Probabilistic NER. Although Probabilistic NER does not make hard decisions when ambiguity is involved, it also does not yet resolve ambiguity. A way of resolving this ambiguity is by using user feedback to let the probabilities converge to the real world situation, called *Targeted Feedback*. The main goal in this project is to improve NER results by using PNER, preventing ambiguity related extraction errors and using Targeted Feedback to reduce ambiguity.

This research project shows that *Recall* values of the PNER results are significantly higher than for regular NER, adding up to improvements over 29%. Using *Targeted Feedback*, both *Precision* and *Recall* approach 100% after full user feedback. For *Targeted Feedback*, both the order in which questions are posed and whether a strategy attempts to learn from the answers of the user provide performance gains. Although PNER shows to have potential, this research project provides insufficient evidence whether PNER is better than regular NER.

Table of Contents

Acknowledgements **i**

Abstract **iii**

Table of Contents **viii**

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 4
- 1.3 Research Questions . . . 4
- 1.4 Research Method . . . 5
- 1.5 Contributions . . . 6
  - 1.5.1 Technical Contributions . . . 6
  - 1.5.2 Societal Contributions . . . 7
- 1.6 Outline . . . 7

2 Background **8**
- 2.1 Named Entity Recognition . . . 8
  - 2.1.1 NER approaches . . . 9
- 2.2 Storage of Probabilistic Data . . . 11
- 2.3 Research Context: NFI . . . 12
  - 2.3.1 NER approach: KEES . . . 12
  - 2.3.2 NER for forensics . . . 13

3 Problem Exploration **14**
- 3.1 NER . . . 14
  - 3.1.1 Domains and adaptability . . . 14
  - 3.1.2 Dutch NER . . . 15
  - 3.1.3 Misspellings and String Similarities . . . 16

3.1.4 NER in Forensics . . . 16
3.2 Ambiguity . . . 16
3.3 Storage of Probabilistic Data . . . 18
3.4 Targeted Feedback . . . 18
3.5 Result Analysis and Tooling . . . 20
3.6 Research Scope . . . 21

4 Probabilistic NER **22**
4.1 PNER Process . . . 22
4.1.1 Extraction Process . . . 22
4.1.2 Usage Process . . . 24
4.2 NER Approach . . . 24
4.2.1 Probability Calculation . . . 26
4.2.2 Entity Probability Distribution . . . 27
4.3 Probabilistic Model . . . 29
4.3.1 Relational Approach . . . 29
4.3.2 A Better Approach: Random Variables . . . 31
4.4 Summary . . . 31

5 Targeted Feedback Strategies **33**
5.1 Targeted Feedback . . . 33
5.2 Strategy Overview . . . 34
5.3 Question Proposal Strategies . . . 34
5.3.1 Naive . . . 35
5.3.2 Random . . . 35
5.3.3 Structural . . . 35
5.3.4 Genius . . . 35
5.3.5 Cluster . . . 36
5.4 Answer Handler Strategies . . . 37
5.4.1 Basic DB . . . 37
5.4.2 Statistical . . . 37
5.5 Discussion . . . 38

6 Experimental Setup **40**
6.1 Method . . . 40
6.1.1 Test and Training Data . . . 41
6.1.2 Strategies . . . 42
6.1.3 Queries . . . 43

6.2 User Simulation 44
6.3 Validation 45
6.3.1 Precision and Recall for Regular NER 47
6.3.2 Strategy Comparison 47
6.4 Subexperiment: NER Approach Comparison 48
6.5 Subexperiment: Entity Probability Distribution 49

7 Results **50**
7.1 Initial Query Performance 50
7.1.1 Regular NER 53
7.2 General Feedback Trends 53
7.2.1 Precision and Recall Climb 54
7.2.2 Recall Jumps 54
7.2.3 Performance Drops 54
7.2.4 Delayed Precision Increase 55
7.3 Strategy Performance 57
7.3.1 Question Proposal Strategies 57
7.3.1.1 Naive and Random 58
7.3.1.2 Structural and Genius 58
7.3.1.3 Cluster 60
7.3.2 Answer Handler Strategies 60
7.3.2.1 BasicDB and Statistical 60

8 Conclusions and Future Work **63**
8.1 Conclusions 63
8.2 Contributions 66
8.2.1 Technical Contributions 66
8.2.2 Societal Contributions 66
8.3 Future Work 67

Bibliography **69**

Acronyms **74**

Glossary **76**

A PNER Framework Implementation **77**
A.1 Data Model 77
A.2 Framework 79

A.2.1 Extraction Process 79
A.2.1.1 Reading 79
A.2.1.2 Extraction 79
A.2.1.3 Ambiguity Analysis 81
A.2.1.4 Rule Engines 83
A.2.1.5 Probabilistic Database 83
A.2.2 Usage and Feedback 84

B PNER Query Language **86**
B.1 Simple Queries 86
B.2 Type Constraints 86
B.3 Relational Queries 87
B.4 Document Constraints 87

C Stanford NER CRF Model Training **88**
C.1 Input: SoNaR Partitioning 88
C.2 CRF Training Parameters 88

D Random Variables Approach **92**
D.1 Random Variables 92
D.2 Example 93
D.3 Reference Ambiguity 95

E Feedback Simulation Results **96**
E.1 Basic Database Strategy 96
E.1.1 Naive Strategy 96
E.1.2 Random Strategy 97
E.1.3 Structural Strategy 98
E.1.4 Genius Strategy 99
E.1.5 Cluster Strategy 100
E.2 Statistical Strategy 101
E.2.1 Naive Strategy 101
E.2.2 Random Strategy 102
E.2.3 Structural Strategy 103
E.2.4 Genius Strategy 104
E.2.5 Cluster Strategy 105

Chapter 1

Introduction

This chapter provides an introduction to this research project in Probabilistic Named Entity Recognition (PNER) and elaborates on the problem statement, research questions, method and the contributions of this research.

1.1 Motivation

In forensics, there is an ongoing shift from physical to digital evidence. Nowadays when a suspect is taken into custody, often multiple data carriers are seized, like personal computers, smartphones, USB devices, etc. These data carriers contain large amounts of mostly unstructured data, ranging from formal documents to visited webpages, chats and email. Another large semi-structured dataset submitted to analysis in forensics is the world wide web.

These large amounts of unstructured data need to be processed in order to find evidence, detect security risks, etc. Performing manual analysis on datasets of such proportion is an immense job. Therefore, Information Extraction (IE), the automatic extraction of information such as entities from unstructured data, is essential [54]. The first step in IE is often the detection and classification of proper names, also called Named Entity Recognition (NER) [34]. Although NER has already been researched intensely, it remains a difficult task [8].

Besides the field of forensics, NER is a useful technique in many different domains, even in everyday life. When opening an email or note on a state of the art smartphone like the iPhone, it will automatically perform NER, creating clickable phone numbers, addresses, dates and events, linking them to the corresponding actions on the phone.

In NER, one of the main challenges is the ambiguity of the recognized named entities [70]. Whether a numerical value represents a phone number or a social security number is called semantic ambiguity and whether the correct boundaries for an entity are 'Lake Como' or 'Como' is called structural ambiguity. Such ambiguity is often quite trivial for a human who can more easily interpret the context [67], but far less trivial for a computerized NER process.

Most NER solutions identify the multiple alternative interpretations of an entity and most machine learning approaches even assign probabilities to these alternatives internally, like e.g. Conditional Random Fields (CRFs) [36,71]. However, after identification, most solutions choose the most probable alternative as the correct answer. For example, choosing a company name with a probability of 0.6 over a name of a person with a probability of 0.4. This does however not make the company name the absolute correct answer, introducing the risk of taking wrong decisions and assigning the wrong type to an entity. In large datasets, these decisions will inevitably result in errors, working through in the following stages of processing the data, like performing data mining or having a forensic investigator query and analyze the data. Working with data containing such extraction errors can slow down the search for evidence or even result in not detecting certain evidence or security risks.

A method to prevent these extraction errors is to postpone the decision or by not making a decision at all. Every identified alternative can then be assigned a confidence score that correlates with the probability of this alternative being the correct one [26]. Not making this decision and taking all identified possible alternatives scored with a probability will be called *Probabilistic Named Entity Recognition (PNER)*.

Using PNER results in an uncertain dataset, resulting in uncertain answers for queries. As Helland et al. [28] state, we can no longer pretend to live in a clean world. A method to reduce this uncertainty is by serving the user questions targeted on reducing ambiguity and uncertainty. Finding the best question to pose to the user and the best way of handling the answer provided by the user will be called *Targeted Feedback*.

Figure 1.1 provides an example of results of PNER, showing a view on a document in the Probabilistic NER Browser, developed as part of this research project. Yellow marked text denotes entities with semantic ambiguity, e.g. `Schaarbeek`. This entity has as possible types location (`94%`) and person (`5%`). Then, orange marked text denotes structural ambiguity, e.g. `René Magritte Onder`. The exact boundaries are unknown, while in fact `Onder` is not part of the

entity, it is considered as an option. Finally, the blue entities denote uncertain entities with only one assigned possible type.

Parsing the example in figure 1.1 using regular NER results in `Brusselse` and `Academie` as two separate entities. This is mainly due to the fact that the highest probable alternative for these entities differ. However, the combined entity of type organisation is the correct answer here. Regular NER missed this option, but PNER denotes this with a probability of `17%`. Performing a query like *'select all organisations that show a relation with the suspect'* with regular NER would not show such an entity, where PNER would. Using Targeted Feedback, the ambiguity of such an entity can be removed, raising the probability of this correct answer.

Fig. 1.1. PNER Browser: Ambiguity in a document

Storing and querying uncertain data can be done using a probabilistic database. Other applications for probabilistic databases include data integration, information retrieval, and management of sensor data [58]. The fields of probabilistic databases and Information Extraction have only recently intersected [27,26,72]. However, research in Probabilistic Named Entity Recognition has not yet been done. Therefore, this research project has as main goal the exploration of the

Probabilistic NER problem. More specifically, using PNER in order to improve NER results by preventing ambiguity related extraction errors and using Targeted Feedback to reduce the ambiguity in the extracted results.

1.2 Problem Statement

The problem of Probabilistic NER consists of many subproblems. This section merely elaborates on the problems that are taken on in this research project. Chapter 3 elaborates on the other problems relevant for Probabilistic NER, but not within the scope of this research project.

How to approach NER in a probabilistic way is the main problem to solve. What a PNER process should look like and which steps are to be incorporated in a PNER process receives the focus of the first part of the research project. A PNER process is designed and a framework is implemented based on this process, which is used for experimentation for the second part.

Storing all possible alternatives results in not throwing away possibly correct answers, it does however not resolve ambiguity. In fact, it results in an explosion of uncertain data. The second part of this research project focuses on how to reduce the proportion of the uncertain data, reducing ambiguity and increasing data quality. However, by creating disambiguation rules or models that automatically throw away alternatives, the type of errors that are now prevented on extraction time would be reintroduced. Instead, Targeted Feedback is used, consulting the user to resolve ambiguity. Two subproblems in Targeted Feedback are finding the best question to pose to the user and how the answers of a user can be used to not only affect the entities in the question, but also probabilities of similar entities outside the context of this question.

1.3 Research Questions

The main goal of this research project is to improve NER results by preventing ambiguity related extraction errors, introduced at extraction time by making hard decisions, and resolving ambiguity using Targeted Feedback. Keeping this goal in mind, the following main research question is formulated:

> *"How does Probabilistic NER in combination with Targeted Feedback compare to regular NER?"*

The answer to this main question will depend on both the starting values of the data quality in comparison with regular NER and the speed of raising the

data quality during user feedback. To come to a structured answer to the main research question, the following subquestions are identified and will be answered accordingly:

R1 *Which subproblems play a role in Probabilistic NER?*
Literature study & analysis of existing software

R2 *What should a Probabilistic NER process look like?*
Literature study & analysis of existing software

R3 *What is the best strategy for finding the best question to pose to the user?*
Research & development and experiments on multiple strategies

R4 *What is the best strategy for learning from the answers given by the user?*
Research & development and experiments on multiple strategies

R5 *How does PNER in combination with Targeted Feedback compare to regular NER?*
Experiments and result analysis

Answering these subquestions will provide enough information to be able to answer the main research question. To be able to get a quick peek on one of these subquestions, section 1.6 also describes in which chapter or section which research question is answered.

1.4 Research Method

This section briefly describes the research method. Chapter 6 goes into more detail on the method and provides motivation for certain decisions.

For the first part of the research project, a literature study is conducted in the fields of NER and probabilistic data(bases). Using this information and performing analysis of existing NER software, the existing problems in the domain of PNER are mapped, resulting in the design of a PNER process. This process is refined by implementing the process with multiple extraction methods.

In the second part of the research project, more experimentation takes place. Multiple strategies for both serving questions to the user and handling the answers provided by the user are developed and tested. This requires an annotated dataset, for which the 1 million words manually annotated part of SoNaR is used [56]. Using this data, the data quality can be calculated using Precision and Recall and more specifically Expected Precision and Expected Recall to cope with probabilistic data. Chapter 6 explains these measures in more detail.

A forensic investigator using NER results will browse through this data using queries like *'select all persons that have a relation with the suspect'* or *'select all frequently occurring locations'*. He is interested in the results of this query and therefore, the feedback questions served to the user should relate to this query, increasing the data quality for that query. Therefore, multiple queries are defined to validate on. By calculating the data quality set out to the number of questions answered, statements can be made on the effectiveness and performance of the Targeted Feedback strategies. The strategy that provides the forensic investigator with the highest quality increase requiring the least number of queries is considered as the best strategy. The performance of the increase is calculated using the area under the F-measure graph set out to the number of questions.

Using both the information on the performance of the Targeted Feedback strategies and the performance of the initial query results, a comparison can be made between PNER and regular NER.

1.5 Contributions

1.5.1 Technical Contributions

As first technical contribution, this research project provides a problem exploration of the field of Probabilistic NER. Using this problem exploration, a formal description of a Probabilistic NER process is provided as second contribution. The third contribution can be found in the implementation of the PNER process, resulting in an open source PNER framework [33].

The fourth contribution of this research project lies in the usage of user feedback to reduce the ambiguity of NER results as well as the explosion of probabilistic data. Several strategies of serving feedback questions to the user as well as handling the answers given by the user are presented. Because no research has yet been performed in this field, this project mainly presents some strategies which can be considered as baseline for future research in this area. Also a few more extensive strategies are provided.

Finally, two new measures are introduced. First, the $E_{100}(Recall)$ measure for measuring the *Recall* of probabilistic data. Second, a measure for comparing Targeted Feedback strategies by calculating the area under the F-measure graphs is presented.

1.5.2 Societal Contributions

As contributions to society, this research project looks into providing forensic investigators with a potentially better approach for finding evidence in criminal cases, detecting risks for public safety, etc. By emphasizing on ambiguity and not throwing away possible correct answers, this approach has a potentially better data quality allowing to speed up the process of finding evidence and detect risks or evidence previously remained unseen.

1.6 Outline

Chapter 2 provides background information on the fields of NER and probabilistic databases. Also background information on the NFI, the institute where this research project has been executed is given in this chapter. Answering subquestion *R1*, chapter 3 provides a problem exploration of the field of Probabilistic NER, concluding with a scope for this research project.

Chapter 4 goes into more detail on Probabilistic NER. This chapter provides the PNER process, information on how this process is filled in and important decisions made while implementing the process, answering subquestion *R2*. Following up, chapter 5 goes into more detail on Targeted Feedback. The implemented strategies for Targeted Feedback which are developed and validated in this project are presented, answering both subquestions *R3* and *R4*.

Chapter 6 describes the method of this research project, containing an experimental setup, more information on the used data and the validation method. Chapter 7 then presents the results of the experiments, providing measurements belonging to subquestions *R3* and *R4*. Finally, chapter 8 provides the conclusions of the research project answering subquestion *R5* and the main research question. A summary of future work can also be found in this chapter.

Additionally, more in depth details are provided in the appendices. Appendix A provides a more detailed elaboration on the implementation of the PNER Process resulting in the PNER Framework. Appendix B provides information on the syntax of the implemented query language. Appendix C describes how the underlying NER method, Stanford NER, is trained and how the input data is partitioned. Appendix D clarifies the Random Variable approach for a probabilistic underlying model, which was not implemented but considered as future work. Finally, Appendix E shows the graphs resulting from the experiments.

Chapter 2

Background

The fields of Information Extraction and probabilistic databases are individually extensively researched, but only intersected recently [27]. This chapter provides a background regarding both fields of research and provides information about the NFI, where this research took place.

2.1 Named Entity Recognition

Named Entity Recognition (NER) is the extraction of named entities from unstructured texts. Jurafsky et al. [34] define a named entity as anything that can be referred to using a proper name. What a proper name is depends on the nature and goal of the application. Common proper names subjected to extraction are names of persons and organisations, locations, numeric expressions like phone numbers, dates, bank accounts, etc. Sekine et al. [57] proposed a hierarchy for named entities within 200 different categories.

In literature, multiple names are used interchangeably to refer to the process of extracting named entities from unstructured texts, namely: Named Entity Recognition (NER), Named Entity Recognition and Classification (NERC), Named Entity Extraction (NEE) and Named Entity Tagging (or NER tagging). Distinction can be made between identifying and classifying the entities using a single stage or using two subsequent stages, respectively named unified and cascaded approaches [25]. In this document, the term NER will be used given the following definition:

"The process of identification, classification and extraction of named entities from unstructured texts."

Extracting useful structured information from unstructured sources is the focus of Information Extraction (IE) [54]. This field has its origin in the Natural Language Processing (NLP) community. Within IE, the recognition and classification of named entities was quickly recognized as one of the important subtasks [47]. Desmet et al. [17] state that it has since evolved into a distinct task which is essential for many fields like Information Extraction, question answering and various other NLP problems.

Figure 2.1 shows screenshots of an iPhone, demonstrating a daily practical use of NER. The iPhone automatically recognizes and classifies the named entities and makes them operable. Tapping the package number will ask to open the UPS tracking website, tapping the address will show the location on the map and tapping the phone number will start a phone call using that number.

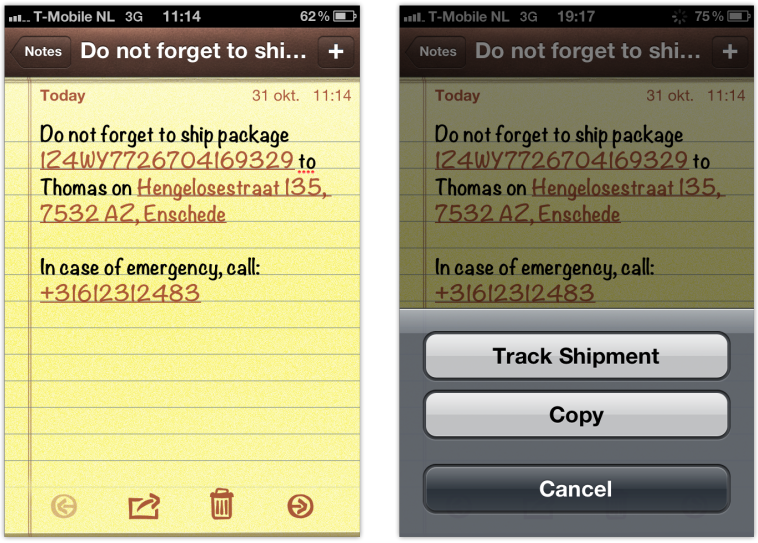

Fig. 2.1. NER on an iPhone, result of tapping the package number

2.1.1 NER approaches

There are a variety of different approaches to implement NER. Most of these approaches can be either summed under rule based, statistical or a hybrid ap-

proach, combining strong points of each approach [8,27,43]. This section provides a description of both the rule based and machine learning approaches.

Rule based NER Most rule based approaches are hand-made systems, using human-made rules to extract named entities. Generally, these systems consist of patterns using grammatical (e.g. part of speech), syntactical (e.g. word precedence) and orthographical features (e.g. capitalization) in combination with dictionaries [43]. Regular expressions are also commonly used.

Especially in a restricted domain, rule based systems can show good performance, having the capability of detecting the more complex entities which are harder to detect for machine learning approaches [43]. However, creating a rule based NER approach is a time consuming and very difficult job [27]. Often, these approaches are domain and language specific and lack portability, robustness and adaptability to new domains [43].

Statistical NER Most recent studies in NER use a statistical, also called machine learning, approach [47]. Finding the most informative features or the optimal settings for a specific algorithm to improve performance is a complex task and quite often receives the focus in research [17].

Commonly used techniques and algorithms are Hidden Markov Models (HMM) [52], Maximum Entropy Models (MEM), Conditional Random Fields (CRF) [75] and Support Vector Machines (SVM). CRFs seem promising [26] and are already found to outperform other techniques like HMMs on a number of real-world labeling tasks [71]. The main advantage of CRFs over HMMs lies in the fact that CRFs are developed to overcome the strong independence assumptions between observations made by HMMs [36].

Three types of machine learning can be distinguished: supervised learning, unsupervised learning and semi-supervised learning. In *supervised learning*, the system reads a large labeled corpus and learns patterns and rules for extraction. A shortcoming of this approach is the requirement of a large labeled corpus. In *semi-supervised learning*, the system is supplied with both labeled and unlabeled data. Rules are constructed from the labeled data, which are then applied on the unlabeled data. The results are used to construct new rules and this process is repeated. When using *unsupervised learning*, the system is provided with only unlabeled data. Lexical resources, patterns and statistics of this unlabeled corpus are exploited [47].

A more recent movement is *active learning*. Like unsupervised learning, unlabeled data is provided to the system. This approach however assumes there is an

all-knowing 'oracle' which can be consulted, the user. Learning is accomplished by asking the user for feedback and combining the learning algorithm with a corresponding interactive learning strategy [31].

Although machine learning seems a better alternative when considering adaptability [62], training a machine learning system on a specific corpus can also negatively influence the adaptability to new domains. A big advantage of machine learning over rule based approaches is the ability to produce the alternative possible labels for an entity, already scored with confidences. This can serve as a starting point for Probabilistic Named Entity Recognition.

2.2 Storage of Probabilistic Data

As mentioned by van Keulen et al. [27], when working with unstructured text where a lot of noise exists, imprecision (or uncertainty) is expected and only recently the fields of Information Extraction and probabilistic databases have intersected. They envision an approach with as one of the main properties the fact that annotations are fundamentally uncertain [66]. The focus in the intersection of both named research fields lies mostly on how to create a probabilistic model for Information Extraction. Several models like the Per row model, the One row model and the Multi row model are suggested [26]. A model using scalable factor graphs to represent uncertainty in a real-world Information Extraction task is presented by Wick [72]. However, van Keulen et al. [27] mention that these existing approaches are still at a preliminary stage and more effort is needed to create reliable and more practical models.

More in general, storing probabilistic data has already been quite extensively researched. In the search for complete data management systems for uncertain data, uncertainty is often incorporated as first class citizen [2,6,55]. Several complete probabilistic database management systems and prototypes are already available, like e.g. MayBMS [45,1,30,37], Trio [59,1,2] and MystiQ [61,9]. Where Trio and MayBMS are open source prototypes for discrete uncertainty, MystiQ also supports continuous uncertainty [64]. Storing annotations resulting from a NER process requires discrete uncertainty. Trio offers as query language an SQL-based language called TriQL, where MayBMS is built on PostgreSQL [51] and simply offers some extensions to SQL provided by PostgreSQL. Performing queries on probabilistic data and performing them in an efficient way has also gotten quite some attention already [15,58,1], but is still a growing field.

Although several probabilistic database management systems are already available, like the earlier named MayBMS, Trio and MystiQ, Wick et al. [72]

state that incorporating probabilities into databases have posed many challenges, forcing systems to sacrifice modeling power. Jampani et al. [32] confirm this by stating that methods which extend relational models can be quite inflexible. Having the representation of uncertainty hard wired into the data model, the types of uncertainty which can be represented are limited.

By manually looking into the existing systems, such limitations were confirmed, e.g. by having to define new tables for every operation on the data and by omitting certain (simple) types of queries due to potential unscalable query execution. A new probabilistic database called MCDB, which is based on a Monte Carlo approach is currently under development [32]. Where extending relational models for probabilistic usage can be quite inflexible, MCDB will be an approach that claims to solve a lot of limitations present in the other approaches [32].

2.3 Research Context: NFI

At the Netherlands Forensic Institute (NFI), the Knowledge- and Expertise Centre for Intelligent Data Analysis (Kecida) has been called to life. Kecida focuses on the task of creating an inventory, evaluating and applying techniques for intelligent data analysis. In this way, Kecida supports the government in efficient processing of large amounts of data for public order and safety [49]. Handling data in a smarter way in order to improve the security of the society is stated as the main goal of Kecida.

By analyzing these large amounts of data, several concrete problems can be solved and questions can be answered. Analysis of social networks can map relationships between people and certain groups. Analyzing financial data can help simplifying the detection of fraud. By using data profiling, intercepted xtc pills can be traced back to their production facility and by using text mining, the important parts can be extracted from large amounts of unstructured texts. This is where NER comes into play.

2.3.1 NER approach: KEES

Kecida Entity Extraction Software (KEES) is the NER approach which is currently under development at Kecida. KEES is a knowledge rule based system, built on techniques like dictionary matching and regular expressions. By performing multiple passes over the data, the entities are extracted and persisted in an SQL database. KEES aims on the extraction of entities from Dutch texts.

Separate from the extraction system, KEES includes a browser for the extracted entities. Using this browser, a forensic investigator can perform analysis on the extracted data by executing queries on the database. For entities in the result set of a query it can be quickly seen in which documents they were mentioned. Also, the user can supply a distance in a number of lines for which two entities should be considered related. Making use of this criteria, related entities can be detected.

Although ambiguity is a comprehensive and difficult problem in NER, KEES does not yet tackle it. KEES can however already detect ambiguity, e.g. when it encounters a number it detects that it is either an ip address, social security number or bank account. Managing this generated ambiguity is a yet to overcome problem for KEES.

2.3.2 NER for forensics

Performing NER in forensics can not simply be compared to performing NER for other applications. Where in other applications and in most research NER is performed on documents from a single domain, having a quite homogeneous dataset, this is mostly not the case in forensics.

For instance, when having to find evidence on a personal computer, all sorts of documents on that computer should be processable, including formal documents, visited web pages, chats, emails, etc. What makes NER so specific in the application domain of forensics is mostly due to this heterogeneous nature of the incoming data, making adaptability to new domains and a cross-domain approach important. Also, texts from an informal domain, like chats, email or SMS, are more common in forensics and have different characteristics from formal texts such as newswire [46].

New domains, like e.g. Twitter [63], are also becoming more interesting for forensics. Recently, news of big events, sometimes including the first photographs, is often first spread on Twitter. Also, recent newswire shows that the government is analyzing Twitter more and more for public safety, carrying out preventive arrests [68]. Analyzing Twitter to detect threats would require a NER implementation that supports streaming data to extract the entities from Twitter in realtime.

Chapter 3

Problem Exploration

This chapter provides a problem exploration of Probabilistic NER. The current challenges are mapped, along with related work on these challenges, concluding with the scope of this research project.

3.1 NER

Although NER has been extensively covered in research, some challenges still remain. This section describes some of these challenges that are relevant in the context in which this research project has been executed.

3.1.1 Domains and adaptability

Creating a domain specific and not very well adaptable system is a risk when developing a NER system. Turmo et al. [62] state that one of the main drawbacks of IE technology is the difficulty of adapting an IE system to a new domain.

Some applications are meant to be domain specific, like e.g. NER for biomedical texts [25], where the applicable named entities are specific objects and formulas referencing e.g. genes and proteins. For other applications, like e.g. forensics, adaptability is of a higher interest. For example, the full contents of a suspect's computer will be a highly heterogeneous dataset, containing documents like chats, emails, formal documents, visited web pages, etc. Extracting entities from such a varying dataset calls for a cross-domain system, adapting easily to

new domains. Ku et al. [38] also state that a successful and useful crime information extraction system should achieve high Precision and Recall regardless of the type and origin of the information.

Some attempts are done in creating an adaptable system, among which are multi-strategy approaches [21], combining multiple learners to achieve better results. Turmo et al. [62] describe multiple adaptive approaches using machine learning techniques in their survey. They state that the current evaluation framework for adaptive IE tasks does not provide sufficient data yet for performing significant comparisons.

In forensics, informal texts like e.g. chats, emails, text messages, web content, etc. are especially interesting. Minkov et al. [46] provide the following definition for informal texts:

> *"Texts that lack the properties of being written for a fairly broad audience and the authors taking care of preparing the document."*

Given this definition, informal texts have different characteristics from formal texts such as newswire. Only little research is done yet on NER in informal domains [50,12,46]. Also, the availability of labeled informal corpora is low at this moment. However, a big Dutch corpus, SoNaR [53,48,11,56], consisting of 500 million words and incorporating texts from 36 domains including several informal domains is just recently released. This corpus could significantly boost Dutch research for informal domains.

3.1.2 Dutch NER

Research in NER for the Dutch language has not gotten much attention yet and has only been studied recently [8,17]. Daelemans et al. [14] recognize this gap and describe the current state of art and define a baseline infrastructure which should be available for Dutch Natural Language Processing (NLP), containing information on NER, corpora, POS taggers, etc.

Some Dutch corpora are available. Most are however not sufficiently large and do not cover a great range of, especially informal, domains. Also, not all corpora are cleared by Intellectual Property Rights laws and can therefore not be used for all purposes [53]. Due to these reasons, the earlier mentioned new Dutch corpus SoNaR has been developed and will be a widely available, large and balanced reference corpus for contemporary written Dutch, containing many different domains, among which are several informal domains.

3.1.3 Misspellings and String Similarities

Using dictionaries to recognize entities is a frequently used technique in NER. However, this assumes that entities are spelled correctly. In informal texts, misspellings, abbreviations and purposely deviant use of language are more common, leaving entities undiscovered when lookups fail due to misspellings. This problem also occurs when analyzing documents subjected to OCR, which is very common for police reports. Ku et al. [38] have used a spell checker in analyzing police reports and found that both Precision and Recall improved slightly.

Coping with misspellings and calculation of string similarities is a field that is already quite extensively researched in the past. Common solutions are edit-distance and Q-grams [4]. However, languages like German and Dutch are morphologically more complex and therefore less trivial and more challenging for correcting spelling than e.g. English [69]. In the specific area of the Dutch language, less information can be found regarding these subjects.

3.1.4 NER in Forensics

The previous sections already uncovered some factors that pose specific challenges for NER in the field of forensics. Although the field of NER is quite extensively researched, it is not yet an exhausted field. It can be concluded that no specific NER system for forensics exists [8] and only little research is done specifically into NER for forensics.

Ku et al. [38] developed an interviewing system which uses Natural Language Processing to extract crime information from sources like police reports, newswire, etc. Chen [12] developed an approach for analyzing Jihadist Dark Web forums, which often have relevance to e.g. Al-Qaeda, providing information on communities and participants.

3.2 Ambiguity

Wacholder et al. [70] state that one of the main challenges in processing natural unstructured text is ambiguity. Three types of ambiguity are mentioned by van Keulen et al. [67]:

- **Structural ambiguity** refers to ambiguity regarding the structure and boundaries of entities, is the word 'Lake' part of the entity for the location 'Lake Como'?

- **Semantic ambiguity** refers to the classification of an entity, does Paris refer to a name or a location?
- **Reference ambiguity** refers to the question to which real world object an entity refers. Does the location Paris refer to the Paris in France or one of the other 158 Paris instances found in GeoNames [23] for cities and villages?

Figure 3.1 provides a translated example from SoNaR, displaying all of the above kinds of ambiguity. Reference ambiguity is represented by linking entities to dictionaries that refer to real world objects, like GeoNames [23]. Structural ambiguity is represented by the choice of one of the boundary combinations for a structural ambiguous entity.

Figure 3.1 also shows interaction between types of ambiguity. If the correct structural boundary is `European Centre Brussels`, then `Brussels` within this entity no longer has reference ambiguity.

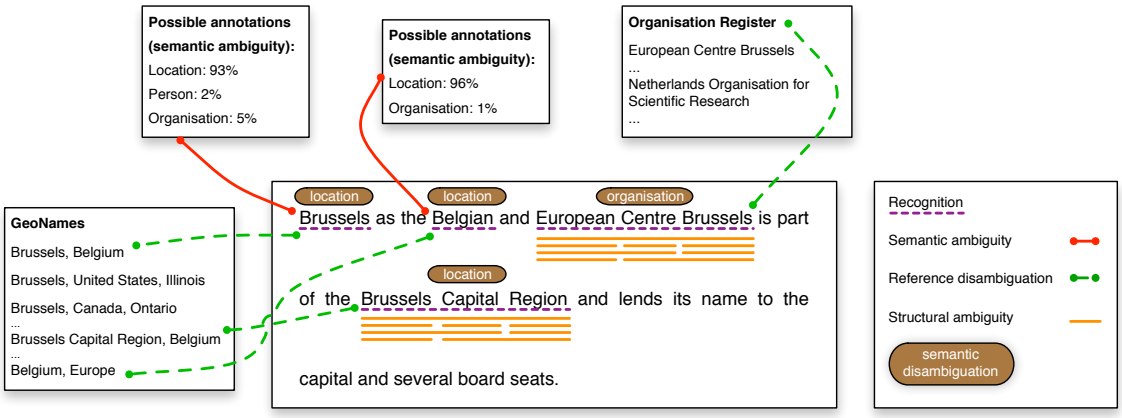

Fig. 3.1. Recognition of entities, Semantic, Structural and Reference ambiguity and disambiguation

Regular NER mostly results in a dataset containing entities for which the structural and semantic decisions have been made. Some tools can identify this ambiguity and perhaps even score the alternative classifications with a probability. However, most current tools simply pick the highest probable classification and therefore not solve such ambiguity, perhaps even introduce errors.

Solving reference ambiguity by establishing the mappings between the entities and the real world objects they refer to is also called named Named En-

tity Disambiguation (NED), entity resolution or Named Entity Normalisation (NEN). Hoffart et al. [29] present a robust method for disambiguation, using the context, knowledge bases and coherence graphs. Their online tool, AIDA [44,74], shows good performance, although in the end it picks the highest probability using a similarity measure, thus still prone to errors [29].

Ambiguity or uncertainty is also present in the fields of Data Integration [39] and Data Fusion [7]. Important tasks here are Schema Matching [41], Schema Mapping [41] and Duplicate Record Detection [19]. In Schema Matching, it is uncertain which fields in the different schemes carry the same semantical information. In Schema Mapping, it is uncertain how to format these matches to get a general, canonical form. Duplicate Record Detection is the task of detecting whether two different records describe the same real world item, also called entity resolution or entity disambiguation. Also in these situations, when an absolute decision is made, errors are introduced. In Probabilistic Data Integration, this problem is solved in a probabilistic way, preserving confidences of whether something matches [65]. In theory, every record could match with each other, leading to massive data explosion. Van Keulen et al. [16] mention that this data explosion problem can be kept in a manageable form by adding simple knowledge rules. Magnani et al. [41] notice that uncertainty management has become recognized as a fundamental aspect of data integration and that it is accepted that uncertainty is relevant and that it might not be possible to remove all uncertainty during integration.

3.3 Storage of Probabilistic Data

When approaching NER in a probabilistic way, the ability to store and query probabilistic data becomes a necessity. Section 2.2 already elaborated on this subject and mentioned that the research field of probabilistic databases has already been extensively researched. However, the fields of IE and probabilistic databases have only recently intersected and existing approaches for a probabilistic model for Information Extraction are still at a preliminary stage [27].

3.4 Targeted Feedback

Preserving all probabilistic data by keeping all alternatives can clutter the results for eventual usage. To improve this usage, it is important to look into how to let the probabilities converge to the real world situation. Creating an algorithm to automatically disambiguate the data will reintroduce the errors that

PNER attempts to reduce. Wacholder et al. [70] state that for the foreseeable future, extraction tasks will require human effort for disambiguation. Although, large-scale extractions can not be performed under *full* user supervision [26]. Also in the field of data integration, Doan et al. [18] state that the task of finding semantic mappings cannot be fully automated and therefore it is crucial to develop tools for assisting the process to achieve large-scale data integration.

Figure 3.2 shows an *ordered* probability distribution for all instances of the entity 'Amsterdam' having location as a possible classification. This graph demonstrates a possible initial ordered distribution and a *reordered* distribution after *complete user feedback*, removing all ambiguity for the case that these instances of 'Amsterdam' are a location or not.

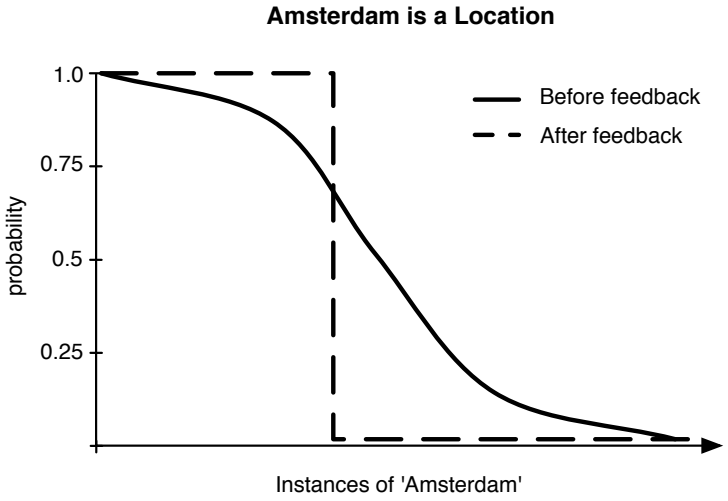

Fig. 3.2. Ordered probability distribution before and after full user feedback

So, serving questions to the user, resulting in feedback for disambiguation is inevitable. The user's time and effort are however costly, so the feedback should be requested in the most efficient and useful way, requiring a quick and painless interaction from the user [31]. Finding the optimal question and the optimal way to treat the feedback of the user will be called *Targeted Feedback*. By requesting feedback on query-time, the user will not be bothered solving disambiguation problems that are not relevant for the current goal envisioned by the user.

Although the user is considered an 'oracle' when using feedback, human performance is in fact not necessarily 100%. A study presented Precision and Recall values for human labelers of respectively 82% and 79% for texts from a domain involving technical micro-electronics [73]. Although the user can exploit the context better than a computerized process, a text is often written for a certain target audience. When the user does not belong to this target audience, he might miss implicit context essential for disambiguation.

Recently, also in Machine Learning, feedback has become of more importance. Active Learning [31] attempts to learn from unlabeled data, with as main distinction from unsupervised learning that it involves the user. The user is asked to label data, from which can then be learned. This approach consists of a learning algorithm and a corresponding interactive learning strategy.

Although several statements are made with regards to feedback, in general it remains an underexposed field of research. Fuhr [22] presented a probabilistic model for uncertain queries and imprecise information in databases. By letting the user rank the results of a query, probabilities and future query performance are improved. Van Keulen et al. [65] have shown that user feedback is effective in gradually improving the integration quality when performing probabilistic data integration. In this approach, feedback is used to remove *impossible* worlds from the data integration result. This research however merely shows that user feedback is effective and does not lay the actual focus on user feedback.

3.5 Result Analysis and Tooling

The ultimate goal for which NER is deployed is mostly extraction of desired information or answering a certain question. To achieve this, techniques like data mining or manual analysis might be performed on the (probabilistic) results.

Using a probabilistic approach to NER results in an explosion of possible entities and annotations, increasing the amount of noise in the result dataset. This increased amount of noise might be troublesome for human analysis, introducing the need for good tooling to assist the user. Resolving ambiguity and raising data quality is therefore important for the user. However, for automated mining tools this might be less of a problem. It could be that the true potential of Probabilistic NER comes to light when deploying such a mining step, where data mining tools can exploit the confidence scores. Connecting the Probabilistic NER approach to data mining tools like Weka [40] can make this easier and provide another way to validate the approach.

3.6 Research Scope

It can be concluded that the field of Probabilistic NER is a new research field and consists of many subproblems, like NER in general, adaptability in NER, Dutch NER, the three types of Ambiguity, storage of probabilistic NER data, feedback, etc. Due to the time available in this research project, not all challenges can be accepted in this project.

Although building a NER system for the Dutch language, adaptable to multiple (informal) domains is important, this challenge is too extensive for the scope of this project and would leave little room for the probabilistic aspect. Also, approaching NER in a probabilistic way introduces ambiguity and therefore noise in the results. Developing a system that stores and queries these probabilistic results in the most efficient way does not decrease this noise.

Initially having a lot of ambiguity in the extracted data might not be a problem for data mining processes, which have the ability to exploit the distribution of probabilities. It might however make human analysis more troublesome. By asking the user for Targeted Feedback, ambiguity can be reduced and the data will more closely resemble the real world. This makes the data more usable for a forensic investigator. Therefore, the main focus of this research project is improving NER results using Probabilistic NER and using Targeted Feedback to reduce ambiguity. In order to find a good way to approach Targeted Feedback and make the extracted data converge to the real world situation, multiple strategies are experimented. Because managing reference ambiguity is quite an intense problem, the focus lies on both *semantic* and *structural* ambiguity.

Although the focus lies mostly on Targeted Feedback, this does not mean that the rest of the subproblems are not taken into account. In fact, in order to build a Probabilistic NER process, most subproblems are essential. Therefore, subproblems like storing probabilistic data are in fact taken into account, but less rigorous. More pragmatic approaches are taken on these subproblems. With this in mind, the developed PNER Framework has been set up in such an abstracted way that replacing components (e.g. a new probabilistic database or a new NER extractor) requires minimum effort.

Chapter 4

Probabilistic NER

This chapter provides more details on Probabilistic NER, elaborating on Probabilistic NER as approached in this research project, describing the derived PNER process, the used NER approach and the underlying probabilistic model.

4.1 PNER Process

This section describes an abstract view on the designed Probabilistic NER Process. More details on this PNER Process can be found in Appendix A. This appendix describes the implementation of this process, which resulted in the PNER Framework which is used for experimentation.

4.1.1 Extraction Process

Looking at current NER software, it can be concluded that most software has a limited way of handling ambiguity. Therefore, such software mainly has a process containing steps for importing input, performing NER and exporting output.

Looking more into the problems regarding PNER, the most important problems are the three forms of ambiguity, structural, semantic and reference ambiguity. Using this information as basis, a general process has been designed for PNER, which is shown in figure 4.1.

As first step in the chain, the desired input for NER should be made accessible. Whether this input is a filesystem or a Twitter [63] stream, this step should be able to support multiple readers reading from multiple sources. Such a

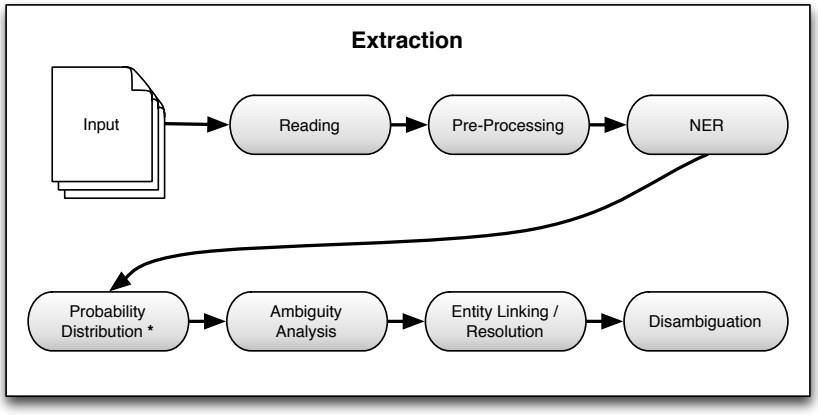

Fig. 4.1. An abstract view on the extraction process for PNER

reader converts the input into a document which may be submitted to different pre-processors. For example, when analyzing webpages from online email clients like e.g. GMail [24], such a pre-processor could strip those HTML pages and extract only the parts that should be submitted to NER.

The data is now ready for extraction. In the NER step, a NER approach is used to extract entities. The next section describes the approach taken in this research project. Depending on the NER approach, the next step is probability distribution. When using a NER approach that does not provide probabilities, this step can be used to assign them.

The following step takes the extracted results, containing ambiguity, and analyzes the data in order to detect both structural and semantical ambiguity. Section A.2.1.3 provides the implemented algorithm for this step. The subsequent step can be used to provide links to Real World Objects (RWOs) and therefore introduce reference ambiguity.

Finally, the extraction process ends with a disambiguation step. Although in PNER, removing entities from the dataset automatically can introduce the errors that PNER attempts to avoid, in some situations it can be useful. For example, some extraction approaches might extract entities of which a human can decide they can never be entities. Also, in this step rules can be made to redistribute probabilities in case the NER approach does not provide a realistic probability distribution.

4.1.2 Usage Process

Once the extraction process has been executed, the results are available to the user. When browsing through this data using the PNER Browser, figure 4.2 shows the associated process figure.

Each time the user executes a query, the Targeted Feedback strategies calculate a question to pose to the user. By answering this question, the user provides feedback which influences the query results. After answering the question, a new question is calculated and posed. This cycle can go on as long as the user is willing to increase the data quality of the extracted results.

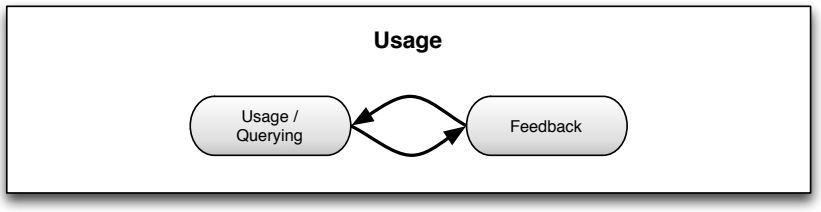

Fig. 4.2. An abstract view on the usage process for PNER

4.2 NER Approach

Developing a NER approach is out of the scope of this research project, therefore an existing NER approach is required. The tool in development at Kecida, KEES, is able to detect ambiguity and was therefore initially to be used. Although KEES does not provide probabilities, it is a starting point. However, while developing the PNER Framework it came to light that KEES only in a small amount of occasions assigned multiple possible annotations to an entity and already makes the hard decisions that PNER attempts to prevent.

Due to this reason, an alternative NER approach was required. The search for a NER approach capable of providing ambiguity and probabilities resulted in either choosing LingPipe [3] or Stanford's CRF Classifier [60,20]. Both LingPipe and Stanford NER work with probabilities and are able to provide them through an API. LingPipe by default uses a Hidden Markov Model (CRFs are also available, but require a lot of effort and tuning [10]) and Stanford NER uses Conditional Random Fields. A sub-experiment has been conducted to be

able to make the choice between these approaches. More information on the experimental setup for this can be found in chapter 6.

Both NER approaches are tested on performance, by training them on the CoNLL 2002 [13] Dutch dataset, using the *train* and *testb* partitions of the data for training and the *testa* partition for validation. Stanford NER has been tested both out of the box without any features and with a set of features they provide in their tutorial (here called Stanford NER+). Also LingPipe is tested out of the box, but with two different so-called chunkers. First, a chunker based on a Hidden Markov Model. Second, a chunker which uses the HMM chunker to generate hypotheses which it then rescores using longer distance character language models. The authors of LingPipe claim that this last chunker is more accurate.

Table 4.1 shows the results of this sub-experiment, displaying the Precision and Recall for the types *Person*, *Location* and *Organisation*. It can be concluded that Stanford NER using a model trained on extended features provides the best overall performance on both *Precision* and *Recall*. Targeted Feedback searches for a question in the extracted results and poses it to the user in order to improve data quality. Therefore, Targeted Feedback can easily resolve *false positives*, but not *false negatives*. Due to this reason, it is important to aim for an initial result with a high Recall value.

NER approach	**P(PER)**	**R(PER)**	**P(LOC)**	**R(LOC)**	**P(ORG)**	**R(ORG)**	**Average**
LingPipe HMM	65.9%	74.1%	74.4%	69.3%	61.2%	54.7%	67.0%
LingPipe Rescoring	65.7%	74.1%	73.4%	**70.4%**	60.9%	**56.6%**	66.9 %
Stanford NER	50.3%	32.6%	77.3%	44.9%	48.1%	30.9%	47.4%
Stanford NER+	***68.4%***	***78.4%***	***78.3%***	*70.1%*	***71.9%***	***56.6%***	***70.6%***

Table 4.1. LingPipe and Stanford NER Performance

LingPipe and Stanford NER are both Java libraries which can easily be used in any Java application. Although LingPipe is able to provide probabilistic results containing ambiguity, a full list of possible entities can not be acquired, only the n-best entities. Therefore, the number of probable entities in a document has to be estimated. Stanford NER provides a probability distribution for

each token in the text and even provides handles for calculation of conditional probabilities regarding previous and following tokens.

Stanford NER provides a better and more usable way of acquiring the probabilistic extraction results and given the experimental setups, shows a better overall performance than LingPipe. Therefore, Stanford NER+ is chosen as starting point for the Probabilistic NER approach.

4.2.1 Probability Calculation

Stanford NER classifies per token, resulting in a probability distribution for each token, as can be seen in figure 4.3.

```
Brussels O=0.0058  ORG=0.1113  MISC=0.0121  PER=0.1371  LOC=0.7012
as       O=0.9840  ORG=7.37E-6 MISC=0.0159  PER=4.36E-6 LOC=2.49E-5
the      O=0.9850  ORG=3.15E-5 MISC=0.0148  PER=6.34E-6 LOC=2.92E-5
Belgian  O=0.9981  ORG=6.22E-4 MISC=8.63E-6 PER=3.53E-4 LOC=8.43E-4
and      O=0.8520  ORG=2.54E-5 MISC=0.1477  PER=2.44E-6 LOC=1.89E-4
European O=1.05E-4 ORG=0.3151  MISC=0.1924  PER=0.2123  LOC=0.2065
Centre   O=5.83E-4 ORG=0.3011  MISC=0.1781  PER=0.1026  LOC=0.2174
Brussels O=1.10E-4 ORG=0.2021  MISC=0.1631  PER=0.2031  LOC=0.3314
...
```

Fig. 4.3. Stanford Classification Result Format

By analyzing the source code of Stanford NER, it can be concluded that the probabilities are discarded when requesting the entities. `Brussels` then corresponds to a location and `as` does not resemble an entity. Entities which span multiple tokens are simply appended while they match classified types. Here, `European Centre` results in an organisation and `Brussels` in a location.

For single-token entities, the probability distribution in figure 4.3 provides the probabilities for that entity being of that type. A threshold of `0.01` is introduced to zero out the zero-probabilities (e.g. `8.43E-4`).

By simply appending follow-up tokens while they match types, Stanford NER disregards the aspect of structural ambiguity. However, using the data in figure 4.3, the possible multi-token entities can be generated. When subsequent tokens are found which have matching types, like `European`, `Centre` and `Brussels`, a sliding window with variable size is used to move the starting point of the entity, generating the possible multi-token entities. Where Stanford NER would discard the multi-token entity `European Centre Brussels`, it is now considered as one

of the options. The probability of these multi-token entities is calculated by exploiting the functionality of Stanford NER to calculate conditional probabilities. Using the *multiplication axiom* [35] for conditional probabilities, as shown in figure 4.4, these probabilities are calculated.

$$P(A_1 A_2 \dots A_n) = P(A_1) \cdot P(A_2 \mid A_1) \cdot \dots \cdot P(A_n \mid A_1 \dots A_{n-1})$$

$$P(\texttt{ European Centre=ORG }) =$$

$$P(\texttt{ European=ORG }) \cdot P(\texttt{ Centre=ORG } \mid \texttt{ European=ORG })$$

Fig. 4.4. Multiplication Axiom (Product Rule) for Conditional Probabilities [35]

This approach of generating structural ambiguity results in a collection of possible entities over a certain boundary. The ambiguity analysis step as discussed in section 4.1.1 converts a stream of entities into such collections, which are further called *entity bundles*.

4.2.2 Entity Probability Distribution

By introducing structural ambiguity as discussed in the previous section, another probability is required. The probability of an entity being a certain type now corresponds to the probability of that entity being of that type multiplied by the probability that this entity exists in the structural aspect. For example, the probability for `European Centre` being an organisation corresponds to the probability that `European Centre` is the correct boundary instead of e.g. `European Centre Brussels` multiplied by the probability that the type of `European Centre` is in fact organisation. Stanford NER does not provide these structural probabilities, therefore they must be estimated.

A sub-experiment has been conducted to find a method for distributing these probabilities. Multiple probability distribution methods have been implemented and tested. More information on the experimental setup for this can be found in chapter 6. Table 4.3 provides an overview of the following methods and their corresponding (Expected) Precision and (Expected) Recall values, which are explained in more detail in chapter 6:

Equal Distribution: Distributes 100% equally as probabilities among structural ambiguous entity bundles.

Probability Mass Distribution: Calculates for every entity in a structurally ambiguous entity bundle the probability mass of the possible annotations and uses this to distribute the 100% among the entities.

Highest Probability Mass Distribution: Similar to the *Probability Mass Distribution*, except that it only takes the highest probable annotation per entity instead of a sum of all possible annotations per entity.

Longest Entity Distribution (x%): Due to the fact that Stanford NER classifies texts per token and when requesting a list of entities simply returns consecutive tokens of the same type, it can be assumed that most of the times this is a correct approach. Therefore, this method takes the longest entity in a structural ambiguous entity bundle and gives it probability `x`, then uses *Highest Probability Mass Distribution* to distribute `100-x`% among the other entities, since this method showed the best overall performance of the above named methods.

Double Token Entity Distribution (x%): Table 4.2 provides an analysis of the Ground Truth of both the *development partition* and the *validation partition* (discussed in chapter 6). It can be concluded that the average entity has a length of ∼1.5 tokens. Using this data and assuming that the average entity has a length of 2 tokens, this method distributes the range of probabilities `x` among the entities with length 2 and `100-x%` to the rest, both methods using the *Highest Probability Mass Distribution*.

Partition	$\#_{Entities}$	**Max length**	**Avg length**
Development	5 899	13	1.56
Validation	6 266	17	1.51

Table 4.2. Ground Truth Entity Length Analysis

The values shown in table 4.3 are calculated over the *development partition* of the SoNaR data, which is explained in more detail in section 6.1.1. The distribution method with the best overall performance is the *Longest Entity Distribution (90%)*. In the following experiments, this distribution method is used.

Method	Precision	Recall	E(Precision)	E(Recall)
Equal Distribution	24.93%	42.48%	50.14%	34.28%
Probability Mass Distribution	24.95%	42.52%	47.98%	37.63%
Highest Probability Mass Distribution	25.06%	42.70%	48.34%	37.86%
Longest Entity Distribution (60%)	37.56%	52.20%	56.15%	37.22%
Longest Entity Distribution (75%)	38.64%	**51.57**%	58.52%	37.70%
Longest Entity Distribution (90%)	***40.77%***	*50.40%*	***61.49%***	***38.51%***
Double Token Entity Distribution (60%)	25.04%	42.69%	53.43%	33.60%
Double Token Entity Distribution (75%)	24.93%	42.45%	56.92%	31.80%
Double Token Entity Distribution (90%)	24.93%	42.53%	61.42%	30.26%

Table 4.3. Entity Probability Distribution on development partition

4.3 Probabilistic Model

Being able to persist and query the data is essential in order to make use of the probabilistic NER results. As mentioned in section 2.2, existing approaches for storage of probabilistic Information Extraction results are still at a preliminary stage. Although essential, this is not the main focus of the research. Therefore, a pragmatic approach towards storing probabilistic data has been implemented. This section describes this approach, its shortcomings and a better approach which was not implemented in this project.

4.3.1 Relational Approach

In order to keep the probabilistic model clearly structured and reduce query complexity, a simple relational approach has been implemented. Entities and annotations are stored in corresponding tables, using inline probabilities. Tables 4.4 and 4.5 provide an example for the text `European Centre Brussels`.

The entities table, table 4.4, contains all possible entities and their boundaries. Structural ambiguous entity bundles can easily be queried by matching the *Parent Boundary*, which corresponds to the boundary of the whole entity bundle. This approach does not require link tables to represent structural ambiguity and therefore reduces query complexity. Each entity is scored with a probability which represents that this entity exists.

Entities

id	*Entity*	*Boundary*	*Parent Boundary*	*Probability*
E	European	(0, 8)	(0, 24)	...
C	Centre	(9, 15)	(0, 24)	...
B	Brussels	(16, 24)	(0, 24)	...
EC	European Centre	(0, 15)	(0, 24)	...
CB	Centre Brussels	(9, 24)	(0, 24)	...
ECB	European Centre Brussels	(0, 24)	(0, 24)	...

Table 4.4. Probabilistic Model: Entities Table

The annotations table, table 4.5, is simply a mapping of entities to annotations. An entity can have multiple possible annotations, scored with a probability, representing the semantic ambiguity of an entity.

Annotations

Entity ID	*Type*	*Probability*
E	ORG	...
E	LOC	...
C	ORG	...
ECB	ORG	...
ECB	LOC	...
...	...	...

Table 4.5. Probabilistic Model: Annotations Table

Due to the fact that this approach was implemented before realizing the requirement of a probability regarding structure, as described in 4.2.2, this is where the actual shortcomings of this approach appear.

This approach does not explicitly model mutual exclusiveness of entities and shifts this responsibility towards the application. This results in the fact that the probabilistic model holds possible worlds like `E` $\wedge$ `ECB`, which are mutual

exclusive. The application can however use the boundaries to detect the overlap of `E` and `ECB` and therefore ignore this option. However, calculation of all the possible combinations of entities within an entity bundle gets quite expensive for bigger bundles and when calculated often.

Due to the fact that mutual exclusiveness is not explicitly modeled and all possible combinations of entities are not available, another shortcoming arises. During probability distribution for entities, described in section 4.2.2, probabilities are distributed among the entities, where they should actually be distributed among the possible combinations of these entities. For example, `EC` and `B` now get probabilities assigned separately, where in fact the situation `EC` $\wedge$ `B` should have been assigned one probability. Although the current approach should reduce query complexity due to its simplicity, this makes querying the most probable combination of entities a lot more complex.

4.3.2 A Better Approach: Random Variables

Due to the shortcomings of the Relational Approach, a new approach is considered, which developed into the Random Variables approach. However, due to time restrictions this approach has been left for future work.

The Random Variables approach considers two random variables, x_1 to represent the semantic ambiguity and x_2 to represent the structural ambiguity. Variable x_1 holds as values the possible permutations of types of the entity bundle, scored with a probability. Variable x_2 holds as values all possible combinations of entities within the bundle, scored with a probability. Multiplying variables x_1 and x_2 results in the probability of a certain entity being of a certain type.

The main difference between the Random Variables approach and the Relational Approach is the way it represents structural ambiguity. Mutual exclusiveness is explicitly modeled and due to the fact that probabilities are assigned to combinations of entities instead of single entities, both shortcomings of the Relational Approach are overcome. More details on the Random Variables approach and an example can be found in Appendix D.

4.4 Summary

The PNER Extraction Process consists of the phases for reading and pre-pocessing input, performing NER and assigning probabilities, detection of ambiguity, introducing reference ambiguity and disambiguation. The Usage Process is an

interaction between the user executing queries and the system posing questions for improvement of the quality of the extracted entities.

For the implementation of the PNER process, Stanford NER is used, providing both easier access to the probabilistic data and better results than LingPipe. By using the per-token probability distribution and the conditional probabilities provided by Stanford NER, structural ambiguity is introduced in the result set. Stanford NER does however not provide probabilities regarding the correct structure of an entity, as they simply append entities with similar types and present them as correct boundaries of an entity. The probability regarding the structure of an entity is introduced by assigning the longest entity in a structural ambiguous entity bundle a probability of 90% and distributing 10% among the other entities in the bundle.

In this research project, the focus is not on the storage of probabilistic data. Due to that reason, a pragmatic relational approach is implemented with inline probabilities. Due to the fact that this approach has a few shortcomings, a better approach is presented, the Random Variable approach, but is not implemented due to the scope of this research project.

Chapter 5

Targeted Feedback Strategies

This chapter elaborates on Targeted Feedback and presents the implemented and validated Targeted Feedback strategies. The chapter concludes with a discussion, sketching strategies for future work.

5.1 Targeted Feedback

Approaching NER in a probabilistic way results in an explosion of ambiguous results, because all possible alternatives are preserved. Although this explosion introduces noise, the data can already be used and queried. Probabilities can be calculated, making it possible to rank answers by likelihood. If the user is unsatisfied with the results or wants to raise data quality, Targeted Feedback can be used. Targeted Feedback reduces ambiguity and raises data quality by consulting the user and not making automated decisions (which again introduce the errors PNER attempts to prevent).

The time and effort of a forensic investigator querying the data are costly. Therefore, the increase of data quality and decrease of ambiguity should happen as fast and early as possible. Having an investigator answer e.g. 1.000 questions before the actual analysis can start would simply not be practical. From this perspective, the practical value of feedback is higher when the maximal increase happens at the beginning and not e.g. after 1.000 questions. The performance of a feedback strategy can therefore be defined by both the speed with which the data quality increases over time and the moment in time of the biggest increase.

5.2 Strategy Overview

Finding the 'best' way to approach Targeted Feedback given the criteria from the previous section is not very trivial. As this chapter will point out, many different strategies towards Targeted Feedback can be found and published research has not yet shown in this field. Therefore this research project performs experiments and presents results on a few baseline strategies and a few more extensive strategies. Some more strategies which were out of the scope of this research project are sketched in section 5.5.

Within Targeted Feedback, two important aspects play a role. First, what question should be posed to the user and second, what can be done with the answer of the user? Can we learn from it and perhaps extrapolate this to other cases? These two aspects result in two different types of strategies, Question Proposal Strategies and Answer Handler Strategies. For Targeted Feedback, one of both is necessary. By splitting strategies in these two strategies, Question Proposal Strategies and Answer Handler Strategies can more simply be joined to see how they perform together. Table 5.1 shows the strategies implemented and validated in this research project. The following sections provide more details on these strategies.

		Question Proposal				
		Naive	**Random**	**Structural**	**Genius**	**Cluster**
Answer Handling	**Basic DB**					
	Statistical					

Table 5.1. Targeted Feedback Strategy Combinations

5.3 Question Proposal Strategies

The Question Proposal Strategies aim on looking through the result set of a query and finding the right question to pose to the user. This section describes the strategies that are implemented and validated in this research project.

5.3.1 Naive

The Naive strategy is the most straightforward strategy. This strategy simply takes the result set of the query, proposing to solve each entity bundle that contains ambiguity in the order of extraction from the initial documents.

5.3.2 Random

The Random strategy does exactly as the name suggests. It first looks for all entities in the result of the query which are ambiguous and then randomly picks one of the entities to pose to the user.

5.3.3 Structural

Having structural ambiguity on top of semantic ambiguity results in a more ambiguous entity. The current approach of handling the results of Stanford NER results in quite some structural ambiguity which can result in big entity bundles, introducing noise in the result set.

Due to these reasons, the suspicion is raised that solving structural ambiguity first might result in raising the data quality at a higher speed and earlier in the process. Therefore, this strategy looks for the structural ambiguous entity bundle which counts the most entities and poses this bundle to the user first.

5.3.4 Genius

The Genius Question Proposal Strategy searches for the most ambiguous entity. When having an entity with two possible annotations with a probability distribution of 95% and 5%, it is not very ambiguous in the sense that the tool was very confident of the annotation with probability 95%. Also, when having a probability distribution of 5%, 2% and 1%, it is quite realistic that this is not an entity at all. As counterpart, entities with probability distributions like e.g. 21%, 32% and 44% are highly ambiguous. This strategy aims on finding entities with such probability distributions.

In order to find these highly ambiguous entities, each entity is assigned an *ambiguity score*. As base of this *ambiguity score*, a quadratic function is used, resulting in a score per annotation probability. This quadratic function is modified in such a way that it's a parabola having the maximal ambiguity at 50%, going towards zero for both 0% and 100%. Figure 5.1 shows the graph that corresponds to this function.

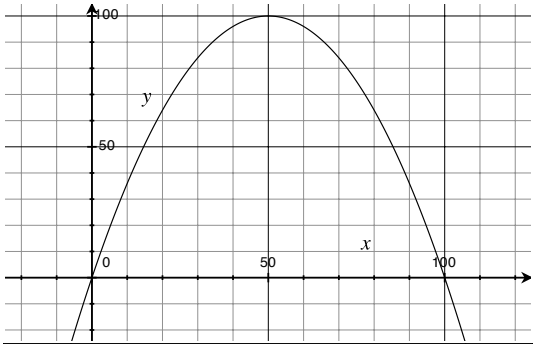

Fig. 5.1. Ambiguity Score Function: $(\frac{-1}{25} \cdot (prob - 50)^2) + 100$

Calculating this score for every possible annotation of an entity and performing the summation over these values results in the *ambiguity score* of an entity. The corresponding formula is shown in Figure 5.2. Each posed question is at that time the entity with the highest *ambiguity score* in the result set.

$$Score(\ entity\ ) = \sum_{an\ \in\ entity} \left(\frac{-1}{25} \cdot (an.prob - 50)^2 \right) + 100$$

Fig. 5.2. Ambiguity Score Calculation

5.3.5 Cluster

When introducing an Answer Handler Strategy that attempts to learn from the answers given by the user, it becomes more interesting to attempt to pose a question which may influence more than one entity. E.g., when we provide feedback saying `Amsterdam` is a location often enough, the conclusion can be drawn that `Amsterdam` being a location has a higher overall probability. From this perspective, a simple clustering approach is used in this strategy.

This strategy clusters entities by the text of the entity, e.g. grouping all entities with text `Amsterdam`. The clusters are ordered descending by the summation of ambiguity scores (see figure 5.2) of the entities within that cluster. From within the cluster with the highest score, the entity with the highest ambiguity score is posed to the user.

5.4 Answer Handler Strategies

When a question is posed to the user and the user answers the question, the Answer Handler Strategies handle these answers. These strategies persist the answer and may attempt to learn from the answer so probabilities can be adjusted on more entities than just the ones present in the posed question. Data quality can be raised by either deleting alternatives or modifying probabilities of alternatives. Important is that the direct answer to the question should be persisted to the database, allowed to delete invalid alternatives. However, when extrapolating to entities outside the entity bundle provided to the user, alternatives may not be deleted, but only probabilities may be adjusted. These strategies are described in this section.

5.4.1 Basic DB

This strategy is the most basic Answer Handler Strategy and only persists the answer of the user in the database. The wrong entities and annotations are deleted and the probability of the correct entity and annotation are increased to 100%. If the correct answer involves a new entity or annotation, they are inserted with a probability of 100%, replacing the initial proposals. All following strategies have this strategy as basis.

5.4.2 Statistical

This strategy extends the Basic DB strategy. After each answer given by the user, this strategy keeps track of a counter per entity / annotation tuple, e.g. (`Amsterdam`, *location*). Using these counters, probabilities for equal entities which have this annotation as possibility can be redistributed.

This first statistical strategy takes a simple approach on how to redistribute the probabilities. For the regarding entities, it multiplies the probability of an annotation with the value of the counter incremented by one. Then, the probabilities are normalized to count up to 100% again. As example, a probability distribution of 60%/20%/20% for which the 60% was the correct answer for another entity results in a distribution of 75%/12.5%/12.5%. Another similar answer results in 81.8%/9.1%/9.1% and so on. The same method is used for redistributing the probabilities for an entity in a bundle.

5.5 Discussion

The field of Targeted Feedback is a new field of research. This research project merely shows the top of the iceberg of possibilities regarding strategies for approaching Targeted Feedback. This section discusses Targeted Feedback strategy approaches that were out of the scope of this research project but might be interesting for future work, resolving limitations of the presented strategies.

Context Analysis The strategies presented in this research project only look at entities, their text and their possible types and do not consider the context like e.g. POS tags of surrounding text. Analyzing the context might be specifically useful for Answer Handler Strategies, extrapolating the given answer not only to entities having the same text, but also entities with similar context.

Feature Vector Deployment In the Statistical strategy, probabilities are redistributed for entities having the same text. Looking forward to the results in chapter 7, this only shows a slight performance improvement in highly heterogenous datasets. Introducing a feature vector could result in a higher performance gain by also redistributing probabilities for entities without similar texts, but with similar features. A feature vector can also be developed to identify ambiguity, which can then be used in a Question Proposal Strategy.

Extrapolation Level Estimation Not all entities can be extrapolated similarly. For instance, the entity `Paris Hilton` is a *highly ambiguous* entity. Paris Hilton can in one context refer to the person Paris Hilton, in another context refer to the Paris Hilton hotel, an organisation, and there is even a fragrance named Paris Hilton, a product. Compared to e.g. `Amsterdam`, which in most cases will be a location, it makes more sense to propose the latter than such highly ambiguous entities.

Extrapolation Scope The current Statistical strategy redistributes probabilities in the whole dataset after given an answer. However, some answers may only apply within a smaller scope. Being able to estimate the scope for which answers have to be extrapolated may boost performance. The user can also be asked to define the extrapolation scope of his answer.

Knowledge Rule Proposal As the user provides more feedback, general trends may be identified. An Answer Handler Strategy can then attempt to find such

trends and propose knowledge rules to the user, which can be used for extrapolation, but may also be used for future extraction. For example, when a user provides feedback for entities `Amsterdam`, `Rotterdam` and `Zaandam` and often annotates them as locations, such a strategy might propose a knowledge rule that entities ending with `dam` are most likely a location. Or perhaps even a hard rule like the single entity `Amsterdam` is always a location.

Probability Redistribution The current Statistical strategy only has one way to redistribute probabilities. When a user provides feedback saying `Amsterdam` is a location, this does not say much yet. However, when a user keeps on providing this feedback, the certainty with which this decision can be extrapolated increases. A redistribution method that more closely resembles this real world process might perform better than the Statistical approach. However, perhaps a more radical redistribution method might show good performance. Multiple redistribution methods can be thought of and tested.

Multiple Entity Proposals The current Question Proposal Strategies all propose one entity bundle per question. Although, providing a few very similar entities to the user to solve at once, this can boost performance. Also, the number of questions is reduced and the learning speed of an accompanying Answer Handler Strategy can be increased. The user should be kept in mind on this, presenting a manageable number of entities.

False Negative Resolution The Statistical strategy only redistributes probabilities after feedback. However, looking at the results in chapter 7, one factor in causing jumps in performance is the resolution of false negatives. Therefore, having an Answer Handler Strategy that introduces new entities or annotations for entities given the feedback might also boost performance. This can also be used in a Question Proposal Strategy, for example in the case of the `Belgi` entity that was supposed to be `België`. A clustering strategy can then use this information to extend or perhaps merge multiple clusters.

Strategy Combinations Some strategies might show good performance in specific situations, but not in general. Looking forward to the results in chapter 7, the Statistical strategy does not show a big improvement on large scale. However, on a smaller scale it shows a big performance improvement. Combining strategies by identifying moments in which the particular strategies provide the best performance boost might lead to outperforming single strategies.

Chapter 6

Experimental Setup

This chapter describes the method used to perform the experiments, elaborating on the validated strategies, the queries used, and test data. It also describes user simulation, describing how the user is simulated during experiments, and which validation measures are used. Finally, it elaborates on two conducted subexperiments, NER approach comparison and entity probability distribution.

6.1 Method

In order to find out how Probabilistic NER in combination with Targeted Feedback compares to regular NER and answer the main research question, experiments are conducted and data regarding the performance of both PNER and Targeted Feedback is produced.

Due to the fact that PNER and NER results are mostly analyzed by querying the extracted entities, various different queries have been composed to validate on, which are presented in more detail in section 6.1.3. The initial performance of these queries can already be used for answering the question how PNER compares to regular NER. However, this does not yet provide information on the combination with Targeted Feedback.

As stated in chapter 5, the 'best' strategy can be defined by both the data quality increase speed and the moment of increase. To find the best strategy among the implemented strategies and be able to reason about the combination of Probabilistic NER with Targeted Feedback, such over-time data quality information should be accessible.

By experimentation with the strategies, such information is generated. The user is simulated, described in section 6.2, and after each question answered by the simulated user, data quality is measured, described in section 6.3. For all the queries and all combinations of the strategies, described in section 6.1.2, this simulation is performed and results in a graph of data quality set out to the number of questions answered by the user. This information combined with the initial performance of the queries are used to see how PNER and NER compare and to answer the main research question.

6.1.1 Test and Training Data

In order to validate the PNER approach, annotated data containing entities and their types, called a *ground truth*, is a necessity. A Dutch corpus containing real world forensic data is not available. Creating such a dataset from scratch involves manual annotation of the texts and is too labor intensive and would leave less room for experimentation. Due to these and following reasons, the existing corpus SoNaR is chosen as working dataset.

SoNaR [53,48,11,56] is a major reference corpus for contemporary written Dutch, recently released in December 2011. It consists of two parts, a 500 million words automatically annotated corpus and a 1 million words manually annotated corpus. Not only the more conventional text types like e.g. newswire, but also new informal text types like SMS, internet fora and email are included.

Kecida is specifically interested in analysis of Dutch texts. The fact that SoNaR is a large sized and up-to-date Dutch corpus, containing contemporary written Dutch, makes it a good candidate for a dataset for this research project. Also, the fact that the 1 million words part is manually annotated means that it is not restricted to the performance of the automatic annotation tool. Due to these reasons, the 1 million words manually annotated part of SoNaR is used for training, development and validation.

Informal text types would be most interesting for the domain of forensics and has not yet been covered in great extend in literature. However, a quick look at the distribution of text types in the 1 million words corpus learns that it mostly contains formal text types like newswire, autocues, wikipedia and brochures. These text types cover the biggest part of the corpus, leaving lesser sized parts of the corpus for more informal types like e.g. websites. Picking one of these smaller sized parts would however lead to partitions containing only a few documents, resulting in a very small data set size to train and validate on. To avoid such a small data set size, one of the bigger parts of the corpus is chosen, the wikipedia

Partition	# words
Training Partition	96 450
Development Partition	75 009
Validation Partition	75 003

Table 6.1. Partitioning of SoNaR wikipedia texts, number of words

text type. Choosing one of the other text types would not have made a great difference due to the lack of differentiation of the text types.

After collecting all the wikipedia text types from the corpus, the texts are partitioned in three parts, one for *training*, one for *development* and one for *validation*. By strictly using the development partition during development, the experiments can be done without a bias or any pre-known knowledge, using the validation partition. Table 6.1 shows a summary of the number of words in either of the partitions and Appendix C provides more detailed information on the exact partitioning of the wikipedia texts.

6.1.2 Strategies

As mentioned in chapter 5, the strategies which are validated can be split into two main categories, Question Proposal and Answer Handler Strategies. Due to the time restrictions of this research project, mostly baseline but also a few more extensive strategies are tested. As recap, table 6.2 presents the strategies which are validated in this research project.

		Question Proposal				
		Naive	**Random**	**Structural**	**Genius**	**Cluster**
Answer Handling	**Basic DB**					
	Statistical					

Table 6.2. Targeted Feedback Strategy Combinations

Because the combination of a Question Proposal and an Answer Handler Strategy together form a Targeted Feedback Strategy, all Question Proposal and Answer Handler Strategies can be combined and validated. Each of these combinations is validated, visible in table 6.2, on the queries which are defined in the following section.

6.1.3 Queries

When a forensic investigator works with the NER results, he or she will browse the results using queries. Therefore, the investigator is interested in the query result and the feedback should aim on improving the data quality of these query results. Serving the investigator questions which are irrelevant to the query will not directly contribute to the data quality of the query result and not directly assist the forensic investigator. Therefore, the feedback strategies aim on finding the questions within the results of the executed query.

By putting the aim on this query-time feedback, the strategies should also be tested and validated on query-time. Therefore, five different queries are prepared to test and validate the strategies on. These queries are selected by browsing and executing queries on the validation partition. The aim was on achieving a varied set of queries in both the aspects of result size, data quality and query type.Table 6.3 provides a summary of the queries which will be validated on and initial performance regarding data quality. More information on the query language can be found in appendix B.

Query	$\#_{en}$	$\#_{an}$	Precision	Recall	E(Precision)	E(Recall)
A: *	28 183	6 597	43.13%	52.17%	61.98%	38.78%
B: *=PER	13 007	4 007	19.17%	54.80%	28.20%	40.35%
C: *=EVE	2 302	925	5.56%	39.76%	14.41%	30.47%
D: *=LOC &2 *Dylan*	635	344	8.72%	81.82%	15.85%	69.56%
E: *=PER &1 Verenigde Staten	160	90	19.79%	90.48%	21.33%	62.33%

Table 6.3. Validation Queries - Performance (en = entities, an = annotations)

6.2 User Simulation

To perform Targeted Feedback, the user is consulted to solve ambiguity in a provided entity bundle which is proposed by a Question Proposal Strategy. Therefore the knowledge and ability to analyze the context of entities of the user are considered as ground truth. Although, in this research project the SoNaR data is used to validate on and the user should consider this as ground truth. Therefore, the user should give the answers as provided by SoNaR.

Due to the fact that such an explicit ground truth is available, the answers supposedly given by the user can be directly derived from SoNaR, introducing the ability to simulate the user. By simulating the user, experiments can be conducted significantly faster and in the given time, more experiments can take place. Feedback can be provided given the following operations:

- Remove an entity (false positive)
- Approve an entity / annotation tuple (true positive)
- Add a new annotation for an entity and approve (false negative)
- Add a new entity within or overlapping the boundaries of the bundle (false negative)

Considering these operations, the simulator simulates the user by performing the following steps for each question:

1. Obtain question in the form of an entity bundle
2. Obtain ground truth for the boundaries of the bundle
3. Approve entity / annotation tuples which match the ground truth
4. Add and approve annotations for existing entities which do not have a matching annotation type with the ground truth
5. Insert entities which are not present in the question but show overlap with the provided entity bundle and are present in the ground truth
6. Remove all entities which are not present in the ground truth

These simulator steps closely resemble the way a real user could provide feedback, provide a statement on the posed entities (approve, remove) or introduce a new entity which should be within these bounds instead of the proposals.

Using this information, the most regular actions on the initial population of entities will be to remove alternatives, either by removing a direct entity or by approving an annotation which removes the alternative annotations. Generally when working with feedback, this entity population will only be reduced, leaving

false negatives undetected. However, by allowing to insert a new entity when none of the proposals were correct, but within that boundary an undiscovered entity (false negative) exists, the initial entity population might occasionally increase in size.

6.3 Validation

Precision & Recall and the F-measure, shown in figure 6.1, are often used measures for NER [34,5] and are also used in forensic research [38]. Using the number of correct answers, system guesses and the total amount of entities in the solution, Precision, Recall and the F-measure are calculated [47].

$$Precision = \frac{\#correct\ entities\ found}{\#total\ entities\ found}\ ;\ Recall = \frac{\#correct\ entities\ found}{\#total\ correct\ entities}$$

$$F\text{-}measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Fig. 6.1. Precision, Recall and the F-measure

Although the F-measure combines Precision and Recall into one score, the Precision and Recall individually provide more meaningful information, which is lost when combining them into one score. In the F-measure, Precision and Recall weigh equally, while it depends on the situation whether the Precision or Recall is more important. Also, Makhoul et al. [42] state that the F-measure exhibits certain undesirable behaviours. Due to these reasons, mainly Precision and Recall are used in this research project.

These measures are developed for certain data, where an entity simply exists or not and an entity has one annotation, and therefore do not account probabilistic data. Since the fields of probabilistic databases and Information Extraction have only recently intersected, a standardized measure for probabilistic NER systems does not yet exist. Defining a threshold before calculating Precision and Recall will measure the performance as if the system introduced the same errors that PNER tries to avoid.

Van Keulen et al. [65] define an expected variant of Precision and Recall specifically for probabilistic data. By taking into account the probability with

which the system claims an answer to be true, the expected value for Precision and Recall can be calculated. Adapting these measures for Probabilistic NER results in the formulas shown in figure 6.2.

$$E(\ \mathit{Precision}\ ) = \frac{E(\ \#\mathit{correct\ entities\ found}\ )}{E(\ \#\mathit{total\ entities\ found}\ )}$$

$$E(\ \mathit{Recall}\ ) = \frac{E(\ \#\mathit{correct\ entities\ found}\ )}{E(\ \#\mathit{total\ correct\ entities}\ )}$$

$$E(\ x\ ) = \sum_{(en,an)\in x} (\ \mathit{en.probability} \cdot \mathit{an.probability}\ )$$

Fig. 6.2. Expected Precision & Expected Recall (en = entity, an = annotation)

In the probabilistic data, entities may have multiple annotations and both entities and annotations have a probability. The probability of an entity existing as a given annotation is the multiplication of both probabilities. So, the expected value of a set of entities corresponds to the summation of the multiplication of the entity and annotation probabilities for every entity / annotation tuple.

Although Expected Precision and Expected Recall are specifically useful for Probabilistic NER, it has not yet been widely used in research. Therefore, both regular Precision and Recall (by introducing a threshold on the highest probability) and Expected Precision and Expected Recall are used to measure the data quality during the experiments.

During user simulation, the Precision and Recall and their expected variants are calculated after each question. Graphs of these values are then plotted to visualize the data quality over time. Using these graphs and the starting values for the validation queries, the main research question can be answered.

Finally, another measure specifically for Recall, $E_{100}(\mathit{Recall})$ is introduced. Because the Recall measures Recall after choosing entities and annotations with the highest probability, this does not show how much of the actual answer is captured in the probabilistic data. Expected Recall does in fact measure all correct answers, but weighs them according their probability, sharing the same problem as Recall. $E_{100}(\mathit{Recall})$ works similar to Expected Recall, but weighs

every good answer as 100%. This way, insight can be provided in how much of the answer is captured in the probabilistic data.

6.3.1 Precision and Recall for Regular NER

Although the *Precision* and *Recall* presented in the previous section take hard decisions based on the highest probability and can therefore be used to represent regular NER, this is not a fair comparison. As section 6.5 explains, Stanford NER does not provide probabilities regarding the structure of an entity. Because these probabilities are estimated, picking the highest probability does no longer represent the performance of regular NER using Stanford NER. Due to these reasons, the *Precision* and *Recall* are also calculated on the results of Stanford NER by not requesting the probabilistic but the regular NER results.

6.3.2 Strategy Comparison

As mentioned in chapter 5, the performance of a strategy can be defined by both the speed of the data quality increase and the moment in time of the biggest increase. No measure yet exists for this purpose, therefore we introduce the area under the strategy graph as measure, as shown in figure 6.3.

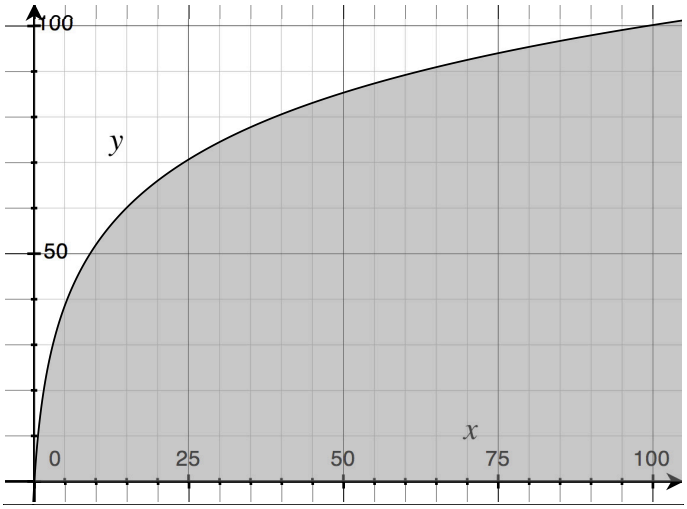

Fig. 6.3. Strategy Comparison Measure - F-measure Area

In order to have one measure on which to compare the performance of strategies, the F-measure for the Precision and Recall is taken. Each point in the

graph represents the F-measure of the data after discarding the less probable alternatives at that given point. Precision and Recall have also been considered separately, but for this particular measure show almost similar results as the F-measure. When there are in fact differences between the performance of Precision and Recall separately considered to the F-measure, these will be named while discussing the results. Using a Riemann integral approach, following the formula in figure 6.4, the area under the F-measure graph is calculated.

$$\frac{\sum_{i=0}^{n} \left(2 \cdot \frac{Precision_i \cdot Recall_i}{Precision_i + Recall_i} \right)}{n \cdot 100}$$

$$n = number\ of\ questions$$

Fig. 6.4. Strategy Comparison Measure

This measure fulfills in comparing strategies, because it acts as expected on both described criteria for strategy performance. If a strategy has a fast speed for increasing data quality, the area increases. Also, if the increase happens earlier in the process, it shows a bigger area than when this happens near the end. Finally, when a strategy actually lowers data quality at a certain point, this measure takes that into account by the fact that this results in a lower area under the graph.

6.4 Subexperiment: NER Approach Comparison

To find the best NER approach which can provide uncertain entities with probabilities, a small subexperiment has been conducted of which more details can be found in section 4.2.

Both LingPipe [3] and Stanford NER [60,20] were options due to the fact that they are both Java frameworks which internally work with probabilities and can be requested by the user of the framework. For both approaches, an out of the box approach and one more extensive approach, e.g. by training on more features, have been tested. Both approaches are validated by calculating the Precision and Recall on the CoNLL 2002 [13] Dutch dataset. SoNaR was not yet available during this experiment.

Also the usability of the framework APIs were tested by working with the APIs and looking more closely into how the probabilistic data is retrieved. Based on both the usability and the performance in terms of Precision and Recall, the best overall approach is taken as basis for this research project.

On both aspects, Stanford NER, trained on more features, has proven to be the best approach. It shows both the best performance and the best usability, offering the ability to retrieve per-token probability distributions and calculation of conditional probabilities.

6.5 Subexperiment: Entity Probability Distribution

As mentioned in section 4.2.2, Stanford NER does not provide probabilities regarding the structure of entities. Therefore, these probabilities have to be estimated before being able to calculate the probability that an entity / annotation tuple exists.

Although this is an important part of the research project, given the time available for this research project this problem should not receive the main focus. Therefore, a subexperiment has been conducted, proposing and validating multiple probability distribution methods. Although more complex distribution methods can be thought of, e.g. by analyzing the context of entities, these more extensive methods have not been implemented and validated due to the earlier named time constraints.

In section 4.2.2, the proposed distribution methods and results are presented. The best strategy is chosen by calculating the Precision, Recall, Expected Precision and Expected Recall on the development partition.

The distribution method that has shown the best overall performance is the *Longest Entity Distribution (90%)*. This method provides a probability of 90% to the longest entity in a structural ambiguous entity bundle and distributes the other 10% among the rest of the entities within the bundle.

Chapter 7

Results

This chapter presents the results of the experiments with the feedback strategies. Due to the number of experiments and number of produced graphs, not all graphs are shown here. In stead, in this chapter the graphs are used to illustrate the results, a full overview of the graphs can be found in Appendix E.

7.1 Initial Query Performance

Because experimentation with probabilistic NER results is quite new, the initial performance of the validation queries can not simply be taken for granted. In order to have a better understanding of the results and to provide insight, this section elaborates on the initial performance, along with the performance of a regular NER implementation.

Looking at the initial performance of the validation queries in table 6.3, the *Recall* and $E(Recall)$ of most queries is rather low. As mentioned in the previous chapter, both *Recall* and $E(Recall)$ do not clearly show how much of the answer is captured in the probabilistic data, respectively by putting a threshold on the highest probability or by weighing the correct answers with their probabilities. Therefore, table 7.1 presents the $E_{100}(Recall)$, showing that the actual *Recall* of the probabilistic data is quite higher than the initially presented results.

Comparing *Precision* and $E(Precision)$ in table 6.3, the $E(Precision)$ value is in all situations higher than *Precision*. So, taking hard decisions based on the highest probability does not weigh up to weighing all correct answers using their probability. In fact, this tells us that a considerable amount of *correct answers*

Query	$E_{100}(Recall)$
A: *	87.33%
B: *=PER	95.59%
C: *=EVE	65.06%
D: *=LOC &2 *Dylan*	95.45%
E: *=PER &1 Verenigde Staten	95.24%

Table 7.1. Validation Queries - Actual Recall

Query	$\#_{en}$	$\#_{an}$	$avg(\ \#_{an})$	$\#_{en}(\ \#_{an} = max\ )$
A: *	26 183	55 381	2.12	760
B: *=PER	13 007	35 951	2.76	760
C: *=EVE	2 302	10 260	4.46	760
D: *=LOC &2 *Dylan*	635	2260	3.56	22
E: *=PER &1 Verenigde Staten	160	508	3.18	9

Table 7.2. Validation Queries - Semantic Ambiguity (en = entities, an = annotations)

in the dataset are not assigned the highest probability among their alternatives and are therefore lost when using regular NER.

Tables 7.2, 7.3 and 7.4 present more information to clarify the initial performance of the queries. Table 7.2 provides statistics regarding semantic ambiguity, for example, query **E** results in 160 entities with 508 possible annotations, an average of 3.18 annotations per entity and there are 9 entities with the maximal number of annotations. Table 7.3 presents statistics regarding structural ambiguity, for example, query **E** has a total of 160 entities within 90 entity bundles having an average bundle size of 1.76, a maximal bundle size of 9 and the ground truth for this query has a total of 21 entities. Finally, table 7.4 provides the average and standard deviation of the probabilities grouped by the annotation type with respect to the whole result set.

Semantic ambiguity plays a big role in the initial performance of the queries. High semantic ambiguity means that the extractor was not very certain of the annotation type of tokens and assigned probabilities to multiple annotation types per token. Within the result set there is a population of 760 entities having a

Query	$\#_{en}$	$\#_{bundles}$	$avg($ $size(bundle)$ $)$	$max($ $size(bundle)$ $)$	$\#_{en\ in\ truth}$
A: *	26 183	6 597	3.97	1 139	6 266
B: *=PER	13 007	4 007	2.35	256	1 699
C: *=EVE	2 302	925	2.49	22	166
D: *=LOC &2 *Dylan*	635	344	1.85	11	44
E: *=PER &1 Verenigde Staten	160	90	1.78	9	21

Table 7.3. Validation Queries - Structural Ambiguity (*en* = entities, *an* = annotations)

Annotation	$avg(prob)$	$\sigma(prob)$
PER	37.79%	37.14
LOC	34.54%	38.88
ORG	19.03%	28.63
PRO	12.85%	21.02
EVE	12.81%	27.67
MISC	8.28%	15.35

Table 7.4. Initial Average Probabilities

possible annotation for every annotation type. Queries **A**, **B** and **C** retrieve all entities or all entities of a certain type, therefore containing this full population. However, while the total number of entities returned by the query drops, this population stays the same, increasing overall semantic ambiguity, affecting mostly the *Precision*.

Structural ambiguity plays another big role and seems to mostly affect *Recall* values. For example, from query **A** to **E** in general, the structural ambiguity is less than the previous query. In the *Recall* values, a reverse trend is visible, having less structural ambiguity results in a higher *Recall* value. Having less possibilities in the structural aspect increases the probability for the correct boundary and the probability of choosing the wrong boundary is lower, therefore increasing the *Recall* value.

Furthermore, query **C** specifically shows very low *Precision* values. Looking at table 7.4, the initial probabilities assigned to entities of type `EVE` are rather low

compared to the more frequently occurring `PER` and `LOC`. Given this information, even when being the correct answer, entities with type `EVE` are assigned a low probability, resulting in lower *Precision* values.

7.1.1 Regular NER

Although the earlier presented *Precision* and *Recall* values take hard decisions, they do not match the results of a regular NER approach, due to the fact that probabilities are estimated regarding the structure of the entity.

Comparing the *Recall* of PNER to regular NER shows in most cases a slightly lower value for PNER than for NER. This difference can be seen in greater extent for the *Precision* values. However, the $E(Precision)$ for query **A** is still higher than the *Precision* for regular NER, meaning that for the total result dataset, performing regular NER does not weigh up to weighing in each correct answer using its probability.

Query	Precision	Recall
A: *	61.61%	57.60%
B: *=PER	64.72%	66.51%
C: *=EVE	88.89%	38.55%
D: *=LOC &2 *Dylan*	31.36%	84.09%
E: *=PER &1 Verenigde Staten	3.19%	95.24%

Table 7.5. Validation Queries - Regular Stanford NER Precision and Recall

Comparing the *Recall* of regular NER to $E_{100}(Recall)$, it can be seen that PNER shows a significantly bigger coverage of the correct answer than regular NER, adding up to a difference over `29%` for queries **A** and **B**.

7.2 General Feedback Trends

The user simulation graphs, provided in appendix E, show some general trends and observations occurring for multiple strategies and queries. This section describes and explains these trends and observations using some of these graphs.

7.2.1 Precision and Recall Climb

A general trend for almost all graphs is that both the *Precision* and *Recall* values approach 100%. Having the *Precision* approach 100% is obvious, because the answer we have is being improved using user feedback. Having *Recall* approach 100% can be partially explained by the initial high *Recall* for the probabilistic data, shown in table 7.1. However, the *Recall* goes even higher. The explanation can be found in step 5 of the user simulation, section 6.2. When an entity is proposed to the user that is incorrect, but it overlaps a correct entity, the user introduces it. For example, the initial results contain errors regarding special characters like ë, resulting in `Belgi` instead of `België`. The user resolves such problems, increasing the *Recall* value slightly more.

Query **C** is an exception on this observation. The *Recall* of this query goes no higher than 71.69%. The explanation for this can be found in the fact that the initial $E_{100}(\mathit{Recall})$ of this query is 65.06%. Introducing overlapping entities does not account for a big enough raise towards 100%. Although, looking at query **A**, the *Recall* for the whole result set reaches 99.41%. This can be explained due to false negatives, in total there were 53 entities which were of type `EVE` but were not assigned this type initially. These entities do not appear in the query result for query **C**, leaving the maximal reachable *Recall* value at 71.69%.

7.2.2 Recall Jumps

Some graphs show quite abrupt jumps in their *Recall* values. Figure 7.1 provides two illustrative examples for these jumps. During the jump in the graph of query **C**, a mixture of events happen that cause this jump. First, from the larger entity bundles presented in this jump, a combination of shorter boundary entities are actually the correct answer. Given the fact that the biggest probability is provided to the longest entity, this raises the *Recall*. Also, in this jump, some new annotations and entities are provided, resolving false negatives.

The jump in query **E** shows an additional cause for these jumps. This jump contains mostly the approval of entities with probabilities around 60% or 20%. After the jump, the approved entities have probabilities of 90+%. Compared to the entities with probabilities around 60% and 20%, the 90+% probable entities only slightly improve *Recall*.

7.2.3 Performance Drops

Although hardly visible in the graphs, there are also drops in performance. For queries **A** and **B**, every Question Proposal Strategy in combination with Ba-

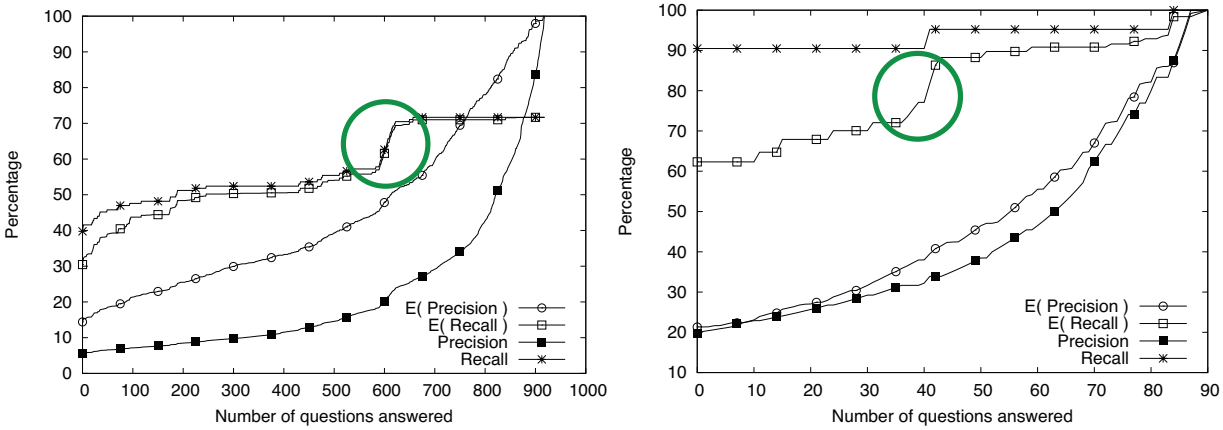

Fig. 7.1. *Recall* Jumps, using BasicDB; Left: C (*=EVE, Naive); Right: E (*=PER &1 Verenigde Staten, Genius)

sicDB shows small drops in the *Precision* values. This happens when for big entity bundles multiple entities are approved. Initially, the longest entity has the highest probability. Now, this single entity is replaced by multiple smaller entities, raising the total number of entities. A higher total number of entities results in a bigger denominator and thereby a lower *Precision* value.

The Statistical strategy also shows drops, which is more logical because probabilities are modified during simulation. The drops are not only restricted to *Precision* values and queries containing big entity bundles. All queries show drops in performance, however always below 1%.

7.2.4 Delayed Precision Increase

Mostly in queries **C**, **D** and **E**, the *Precision* values increase slowly at the start of the simulation and nearing the end this speed rapidly increases. During the rapid increase, no specific events occur compared to the start of the simulation. In fact, the reason can be found in a characteristic of the *Precision* measure.

The *Precision* value is calculated by dividing the number of correct entities by the total entities found. At the start, the population of total entities found is quite high, decreasing while the simulation continues. As the total number of found entities decreases, the number of correct entities increases, increasing in significance. This results in a higher speed at the end of the simulation compared to the start. This can clearly be seen in the graphs of queries **C**, **D** and **E** with the Structural strategy, shown in section E.1.3. Although resolving the biggest

entity bundles first, decreasing the total number of entities, the same delayed increase of the Precision value occurs.

This described effect is demonstrated in figure 7.2 where both the numerator and denominator of the *Precision* fraction are plotted out against the number of questions answered. As the number of questions answered grows, the correct number of entities raises steadily and the total number of entities found decreases rapidly, making the increase of the correct entities grow in significance. As a result, the Precision value increases more rapidly while approaching the end of the simulation.

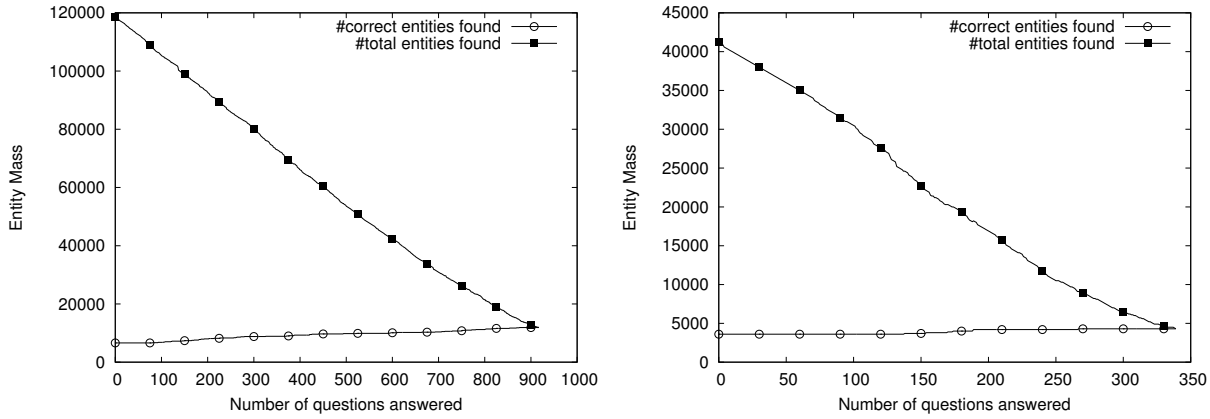

Fig. 7.2. *Precision* fraction, using BasicDB; Left: C (*=EVE, Cluster); Right: D (*=Loc &2 *Dylan*, Cluster)

This phenomenon occurs to greater or lesser extent among the different queries. For example, query **B** shows this behaviour in a lesser extent than queries **C** and **D**. Query **A** hardly shows this behaviour at all. The reason for this can be found in the initial amount of noise (false negatives) in the query results. More noise results in a higher total number of entities and less significance of raising the number of correct entities. A lower *Precision* value corresponds to a higher amount of noise. This can be backed up by looking at the average number of annotations per entity in table 7.2 and the number of entities compared to the number of entities in the ground truth in table in 7.3. For example, query **A** has on average 2.12 annotations per entity and 4.18 times more entities than the ground truth, where these values for query **C** respectively correspond to 4.46 and 13.87.

Like figure 7.2, figure 7.3 shows a similar plot for query **A**. The increase and decrease of the curves are far more equal here, resulting in a steadier overall significance of the increase of the number of correct entities.

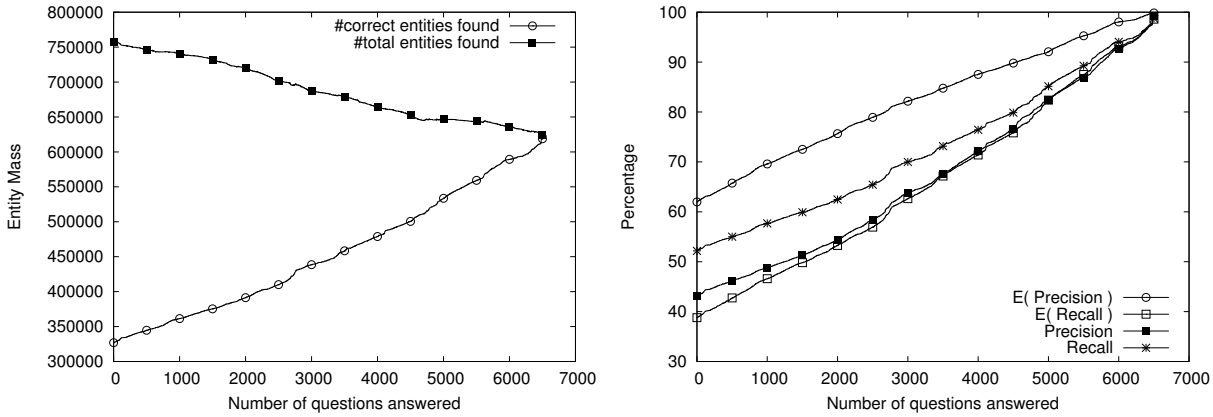

Fig. 7.3. Left: *Precision* fraction, Query A (*, Naive) using BasicDB, Right: Performance of Query A (*, Naive)

7.3 Strategy Performance

In order to compare the different strategies with each other, a measure is introduced that calculates the area under the F-measure graph. More information on this measure can be found in chapter 6. Table 7.6 shows the results for applying this measure on the strategy results.

Looking at the Question Proposal Strategies in combination with the BasicDB strategy in table 7.6, the Structural strategy shows the best overall performance. For the strategies in combination with the Statistical strategy, there is a shift from the Structural to the Genius strategy. For the Answer Handler Strategies, the Statistical strategy on most occasions outperforms the BasicDB strategy. The following sections provide a more detailed analysis of this comparison, accompanied with illustrative graphs to show certain observations.

7.3.1 Question Proposal Strategies

This section provides an analysis of the results on how the different Question Proposal Strategies compare to each other.

		Queries					
		A	**B**	**C**	**D**	**E**	**Avg**
BasicDB	Naive	69.93%	53.39%	29.22%	36.90%	**57.98%**	49.48%
	Random	79.12%	61.79%	32.21%	38.87%	57.17%	53.83%
	Structural	***81.08%***	***63.09%***	***32.94%***	*41.77%*	*57.49%*	***55.27%***
	Genius	80.88%	62.03%	32.42%	**42.13%**	57.95%	55.08%
	Cluster	78.98%	60.63%	31.60%	40.82%	57.18%	53.84%
Statistical	Naive	70.07%	53.56%	29.24%	36.91%	57.98%	49.52%
	Random	81.00%	65.01%	32.03%	39.48%	57.77%	55.06%
	Structural	82.40%	65.34%	**33.17%**	41.78%	57.49%	56.04%
	Genius	***83.37%***	***65.60%***	*32.74%*	*41.93%*	***58.27%***	***56.38%***
	Cluster	81.59%	63.69%	31.90%	**42.07%**	57.44%	55.34%

Table 7.6. Strategy Result Matrix - F-measure Area

7.3.1.1 Naive and Random

Although the Naive and Random strategies are both introduced as baseline strategies, the Random strategy, surprisingly, shows rather good performance in most situations, outperforming the Naive strategy. Figure 7.4 provides two illustrative examples for this observation.

Although the difference in performance is not equally high for all situations, it can be concluded that the order in which the entities are extracted from the texts, in this situation, represents an unfortunate order. With query **E** as only exception, the Random strategy consistently outperforms the Naive strategy. This difference in performance directly demonstrates the importance of looking into the order in which to pose questions. Having an unfortunate order can decrease performance, as is seen for the Naive strategy.

7.3.1.2 Structural and Genius

Where the Random strategy outperforms the Naive strategy by chance, the Structural and Genius strategies outperform the Naive strategy and present the questions in the same order each run. With query **E** as only exception, both the Genius and Structural strategy also outperform the Random strategy

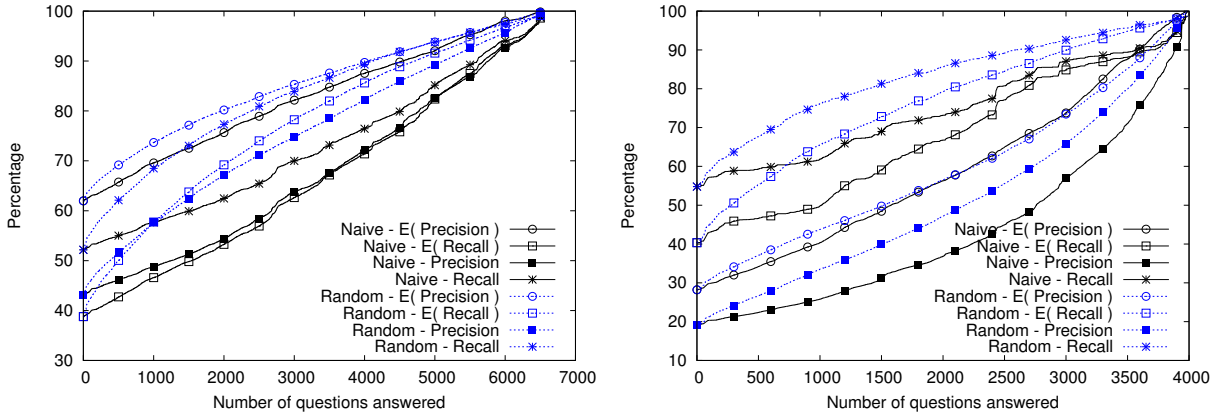

Fig. 7.4. Naive vs. Random; Left: A (*, BasicDB); Right: B(*=PER, BasicDB)

consistently. Figure 7.5 shows the performance of both the Structural and Genius strategy compared to the Random strategy, outperforming the latter.

As can be seen in table 7.6, the Structural strategy shows the best performance in combination with the BasicDB strategy and the Genius strategy shows the best performance in combination with the Statistical strategy. This behaviour is also mostly similar when calculating the area measure for Precision in Recall individually. However, for the performance of the Precision, the Genius strategy outperforms the Structural strategy in combination with BasicDB for all queries except for query **A**.

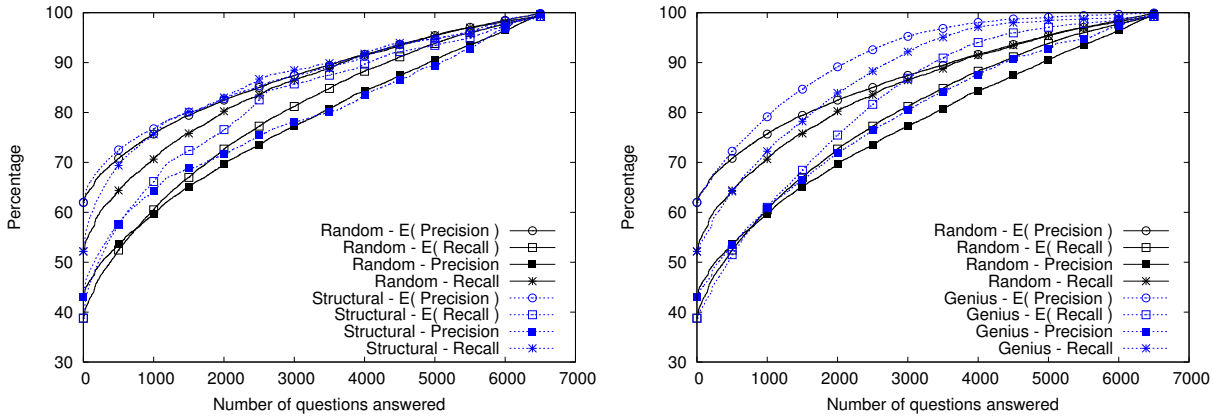

Fig. 7.5. Query A (*, Statistical); Left: Random vs. Structural; Right: Random vs. Genius

Although the Genius and Structural strategies show quite similar performance, differing 0.19% for BasicDB and 0.34% for Statistical, the order in which they pose questions is not similar at all. For instance, in query **A** in combination with the Statistical strategy, the Genius strategy solves the biggest entity bundle at question 202. Also, the Structural strategy starts solving entity bundles of size 1 at question 2609, where the Genius strategy regularly solves such bundles, starting at question 6.

So, although solving structural ambiguity first results in good performance, it does not mean that the structural ambiguity comprise the biggest performance loss in the initial probabilistic data.

7.3.1.3 Cluster

Although the Cluster strategy is introduced mainly as enhancement of the Genius strategy in combination with the Statistical strategy, the Cluster strategy is clearly outperformed by both the Structural and Genius strategy, except for query **D** in combination with the Statistical strategy. So, given the current Answer Handler Strategies, clustering purely on entity text does not introduce a big performance gain. The reason for this can be found in the fact that the query results are highly heterogeneous, as will be explained in the following section.

7.3.2 Answer Handler Strategies

7.3.2.1 BasicDB and Statistical

Although the Statistical Answer Handler Strategy shows better performance than the BasicDB strategy, it is only a slight improvement and less than expected. Figure 7.6 shows two illustrative examples of this slight improvement.

Upon further analysis, it can be concluded that the Statistical strategy does not quite collaborate with the defined queries. The reason for this can be found in the heterogeneous nature of the query results. Table 7.7 shows this heterogeneity by the number of entities and the number of occurrences of the top 5 most occurring entities in these queries.

Table 7.7 shows us that the average biggest cluster in query **A** represents 0.3% of the total number of entities in the query. Trying to improve on a mass of 0.3% will not have a great effect. Although for query **D** the average mass lies at 4.3%, this is mainly due to the biggest cluster, removing this one from the equation makes the average drop to 1.5%.

Given this information, it can be concluded that the Statistical strategy is too naive. Merely keeping statistics for entity / annotation tuples can not cope with

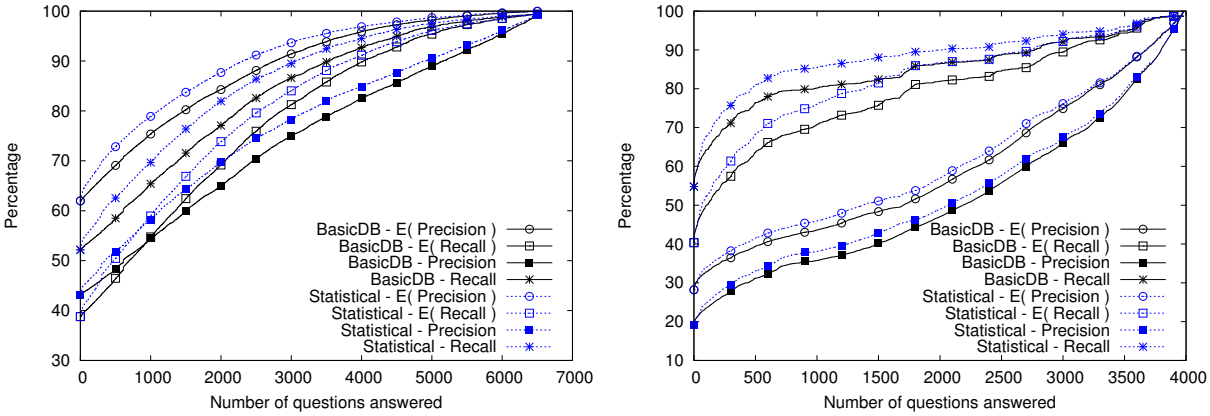

Fig. 7.6. BasicDB vs. Statistical; Left: A (*, Cluster); Right: B (*=PER, Structural)

Query	$\#_{entities}$	**#1**	**#2**	**#3**	**#4**	**#5**	$avg(\%)$
A: *	26 183	90	86	80	77	74	0.3%
B: *=PER	13 007	90	80	74	64	60	0.6%
C: *=EVE	2 302	50	37	32	32	30	1.6%
D: *=LOC &2 *Dylan*	635	87	23	11	9	6	4.3%
E: *=PER &1 Verenigde Staten	160	6	5	3	3	3	2.5%

Table 7.7. Clusters in Validation Queries

results of heterogeneous results. However, this does not imply that the Statistical approach shows equal performance in all situations. Two additional queries have been validated, comparing BasicDB and Statistical using the Cluster strategy. Figure 7.7 shows the results for the queries that return all occurrences of *Dylan* and all occurrences of *Sri Lanka*. With these more homogeneous query results, the Statistical strategy shows much more improvement with respect to BasicDB. Looking at the *Sri Lanka* query, after answering 30 questions, taking the highest probability already results in the maximal possible data quality.

Given figure 7.7, it can be concluded that in certain situations the Statistical strategy shows quite a good performance. While resolving a heterogeneous resultset, at some point in time, this strategy might also provide good performance. In fact, what figure 7.7 shows is what actually happens on small scale. However, this strategy alone does not provide much of an improvement, as seen in 7.6.

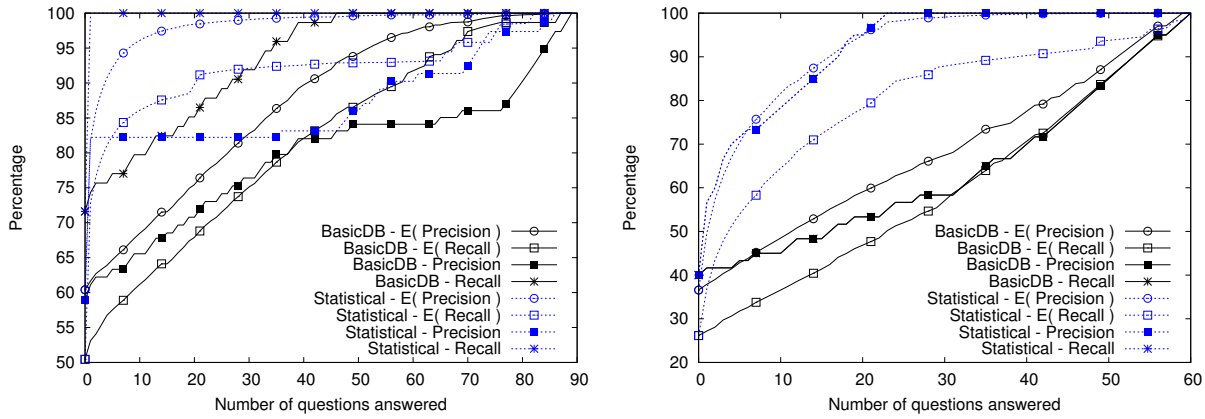

Fig. 7.7. BasicDB vs. Statistical; Left: (Dylan, Cluster); Right: (Sri Lanka, Cluster)

Therefore, combining this strategy with other strategies might provide a better performance with more heterogeneous results, kicking in when the Statistical strategy works at its best.

Chapter 8

Conclusions and Future Work

This chapter provides the conclusions, answering the presented research questions, elaborates on the contributions and presents the future work.

8.1 Conclusions

This section presents the conclusions by answering the research questions as posed in chapter 1, starting off with subquestion $R1$:

R1 *Which subproblems play a role in Probabilistic NER?*

From chapter 3, it can be concluded that the field of Probabilistic NER is a new field or research consisting of many subproblems. The problem of NER itself remains a difficult task [8], having adaptability to other domains and especially informal texts as important subproblems. Furthermore, NER for the Dutch language and for forensics have not gotten a lot of attention yet.

One of the main challenges in NER is the ambiguity [70], subdivided in the three types *semantic*, *structural* and *reference ambiguity*. Having ambiguous Probabilistic NER results, the problem of storing and querying probabilistic data arises. Being able to reduce ambiguity, *Targeted Feedback* becomes an important subproblem by posing targeted questions to the user. Finally, analysis of the probabilistic results and tooling to support e.g. forensic investigators becomes an important subproblem.

Using Probabilistic NER in order to improve NER results by preventing ambiguity related extraction errors and using Targeted Feedback to reduce both *semantic* and *structural ambiguity* is the main goal for this research project.

R2 *What should a Probabilistic NER process look like?*

As chapter 4 has shown, a Probabilistic NER process should exist of multiple steps. A reading and pre-processing step for importing data from various sources and pre-processing the data in the desired format. Then, the NER step performs the actual entity recognition. Depending on the NER approach, a probability distribution step follows, for calculation and distribution of probabilities. Then, the NER results are submitted to ambiguity analysis, detecting both *semantic* and *structural ambiguity*. The subsequent step can be used for introducing *reference ambiguity* and finally, a disambiguation step for removing false positives or redistribution of probabilities.

When the Probabilistic NER results are extracted, the results can be used through the PNER Browser. Using the Targeted Feedback function, the Browser poses questions in order to raise data quality and reduce ambiguity.

R3 *What is the best strategy for finding the best question to pose to the user?*

R4 *What is the best strategy for learning from the answers given by the user?*

To visualize the performance of feedback strategies, the data quality is set out in terms of Precision, Recall, Expected Precision and Expected Recall against the number of questions answered. In order to be able to compare strategies, a new measure is introduced which measures the area under the F-measure graph.

In combination with the BasicDB strategy, the Structural strategy, posing the biggest entity bundles first, shows the best performance. In combination with the Statistical strategy, the Genius strategy, sorting entities using an ambiguity score, shows the best performance. Given the fact that the Statistical Answer Handler Strategy, improving probabilities for entities with similar text, outperforms the BasicDB strategy, only persisting the provided answer, the combination of the Genius and Statistical strategies shows the best performance.

Comparing the Question Proposal Strategies, it can be concluded that the order in which questions are posed to the user matters. For example, the Naive strategy shows a poor order, outperformed even by the Random strategy, resulting in performance gaps over 10%.

Furthermore, it can be concluded that learning from the answers provided by the user and adjusting probabilities outside the query is useful. Although the Statistical strategy has a quite naive implementation, not very well suited for the heterogeneity of the data, improvements in performance are clearly visible.

Using the Statistical strategy on smaller and more homogeneous queries, far better performance improvements opposed to the BasicDB strategy can be seen, showing the importance of finding an Answer Handler Strategy that is able to cope with the heterogeneity of the query results.

R5 *How does PNER in combination with Targeted Feedback compare to regular NER?*

Answering this question answers both subquestion R5 and the main research question. Although PNER can be compared with regular NER, it is hard to say which one is better, because this depends on both the use and user.

Comparing *Recall* for regular NER to the $E_{100}(Recall)$ of PNER, it can be concluded that PNER shows a significantly bigger coverage of the correct answer, adding up to a difference over 29% for the whole dataset. Comparing regular NER *Precision* and $E(Precision)$ of PNER for the whole dataset, it shows that regular NER does not weigh up to weighing each correct answer by its probability. This tells us that a considerable amount of correct answers are not assigned the highest probability and are lost when using regular NER.

Although the probabilistic data covers more of the correct answer and this represents an improvement, this does not necessarily signify that PNER is better. PNER results also contain more wrong answers and ambiguity. Looking at the ambiguity statistics at tables 7.2 and 7.3, this can be confirmed. Also, the fact that for each query the $E(Recall)$ is lower than the *Recall* confirms this.

Targeted Feedback aims on solving this problem by posing the user targeted questions and attempting to learn from the answers provided by the user. When deploying Targeted Feedback, eventually both *Precision* and *Recall* approach 100%, proving its relevance. However, the time and effort of the user are costly. If the user has to solve each found entity before the desired data quality is reached, the user might rather choose to perform NER manually.

The implemented Targeted Feedback strategies do not show spectacular performance yet, but are also only a first step. This research does not provide sufficient evidence that spectacular performance as shown in figure 7.7 can be achieved on larger and more heterogeneous query results. However, it can be said that the combination of PNER with Targeted Feedback has potential.

As mentioned earlier, whether PNER is actually better than regular NER depends on the use and user. There is a significant amount of ambiguity, and therefore noise, in the results. A data mining process might be able to exploit the probabilities and the noise, but a user might find it harder to find what he is

looking for. Although Targeted Feedback reduces ambiguity and increases data quality, one user might be satisfied after answering 10 questions, where another user might answer 100 before being satisfied. Before being able to say whether PNER or regular NER is better, more research is required.

So, it can be concluded that PNER in combination with Targeted Feedback shows real potential compared to regular NER. The initial PNER results cover significantly more correct answers, which would be discarded during regular NER, and using Targeted Feedback, the introduced ambiguity can be resolved and data quality in terms of both *Recall* and *Precision* approach 100%.

8.2 Contributions

This section provides a recap, presenting both the technical and the societal contributions of this research project.

8.2.1 Technical Contributions

This research project provided a problem exploration of the field of Probabilistic NER, found in chapter 3 and converted to future work in the following section. A formal description of a PNER process is provided and as third contribution, this process has been implemented in an open source PNER framework [33].

Although future work is necessary, this research project provided experimental results for Targeted Feedback. It is proven that the order in which questions are posed to the user matters and that learning and adjusting probabilities of entities outside the posed question can result in better performance.

This research project introduced two new performance measures. First, the F-measure area to calculate the performance of Targeted Feedback strategies and second the $E_{100}(\mathit{Recall})$ for the answer coverage of probabilistic data.

In general, this research project has shown that the combination of PNER with Targeted Feedback shows potential compared to regular NER. It is proven that the coverage of correct answers for PNER is significantly higher than regular NER. Furthermore, using Targeted Feedback, data quality approaching 100% can be achieved at the expense of the user resolving all ambiguity.

8.2.2 Societal Contributions

This research does not provide sufficient evidence to conclude whether PNER is better than regular NER. Also, it can not yet be concluded whether the process

of finding evidence is improved. More research is necessary in order to come to such conclusions.

Interesting for the field of forensics is the coverage of correct answers. It is proven that the PNER results show a significantly higher coverage of the correct answers than regular NER, which goes up to a difference of 29%. For forensic investigators, this means that using PNER, answers are found which were previously unseen using regular NER. This might lead to finding evidence or detecting risks previously unseen.

8.3 Future Work

As mentioned in section 3.6, some challenges were out of the scope of this project. For instance, building a NER approach aimed on Dutch text, informal text, adaptability and forensics. Of the ambiguity types, Reference Ambiguity has not been taken on in this project. As probabilistic model, a pragmatic approach has been chosen, where e.g. the Random Variables approach might be a better approach. All of these subjects are considered future work.

The implemented PNER process in this research project uses Stanford NER [60,20]. As discussed in section 3.1.1, multiple learners can be combined to achieve better adaptability. For instance, when Stanford NER shows less performance on a certain type of text, LingPipe [3] might show better performance. Future research can be done in using output of multiple tools in the PNER process. Besides providing more information, covering false negatives, this can also be used to refine probabilities. Also, combinations are not restricted to tools that support probabilistic results, combination of the results of multiple tools that do not provide probabilities can be a start of the probabilistic result.

Resolving ambiguity for probabilistic NER results using Targeted Feedback is a new field of research. This research project merely introduced a few simple strategies, including baseline strategies for further research. This is however only the tip of the iceberg. Section 5.5 sketches some strategies that were out of the scope of this research project and can be seen as starting point for future work regarding Targeted Feedback Strategies.

Often, it will be desirable to process data for which there is no real training corpus available. Therefore, it might be interesting to look into using unlabeled data as starting point. All individual tokens can be given an equal probability distribution and then experiments can be run on how Targeted Feedback can be used to get a streamlined result dataset. The result of such a process might then be used to train on and provide a model for similar data in the future.

This leads to another point of future work. Within this research project, extraction is performed once and Targeted Feedback then improves the initial extraction result. Having Targeted Feedback affect the extraction process, coping with false negatives and initial probability distributions can be improved for new extractions. Implementing streaming data support and improving the extraction phase with feedback might show a great use for monitoring e.g. Twitter.

When resolving ambiguity, not only entities within the query are influenced. By providing entity bundles and by correcting an entity to something outside the query results in a quality raise not only within the query. Entities can also simply be found within multiple queries. This project measured the data quality regarding the query. For future work, it might be interesting to also look at data quality outside the query and how increasing data quality for one query influences the data quality for other queries. It can even be used as a target for a strategy that aims on not only raising the data quality within the query.

As mentioned multiple times, using a probabilistic approach to NER results in quite some noise. Assumed is that forensic investigators querying the data have more hinder from this noise than a data mining process. This is however not proven and not experimented in this project. User studies can be applied to find out how much hinder the user actually has from the generated noise. Also, research into performing data mining on these probabilistic NER results should be performed to find out whether the noise is in fact of less importance and whether it hinders or introduces a new dimension that leads to better results.

Finally, the tooling is important too. Currently, a browser is implemented for visualization of the results and querying the probabilistic NER data. However, the current query engine aims on finding all entities that correspond to the given query and returns all entities. For example, when 20 000 entities are found, they are also displayed and no sorting takes place. By deploying Information Retrieval techniques, the user experience can be improved and might result in a tool where the ambiguity and noise are not at all disruptive for the user.

Bibliography

1. C. C. Aggarwal. Managing and Mining Uncertain Data. Springer, 2009.
2. P. Agrawal, O. Benjelloun, A. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A System for Data, Uncertainty, and Lineage. In Proceedings of the 32nd international conference on Very large data bases, 2006.
3. Alias-i. Lingpipe. Available at: `http://alias-i.com/lingpipe/`, Last visited March 2012.
4. R. C. Angell, G. E. Freund, and P. Willett. Automatic spelling correction using a trigram similarity measure. Information Processing & Management, 1983.
5. R. Baeza-Yates, B. Ribeiro-Neto, et al. Modern Information Retrieval. ACM press New York, 1999.
6. D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. IEEE Transactions on Knowledge and Data Engineering, 4(5), 1992.
7. J. Bleiholder and F. Naumann. Data Fusion. ACM Computing Surveys (CSUR), 41(1), 2008.
8. D. F. Bon. Domain specific named entity recognition for forensic intelligence. Universiteit van Amsterdam, 2011.
9. J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. MYSTIQ: a system for finding more answers by using probabilities. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 891–893, 2005.
10. B. Carpenter. Why do you Hate CRFs? Available at: `http://lingpipe-blog.com/2006/11/22/why-do-you-hate-crfs/`, Last visited March 2011.
11. Centre for Language and Speech Technology. SoNaR: STEVIN Nederlandstalig Referentiecorpus. Available at: `http://lands.let.ru.nl/projects/SoNaR/`, Last visited October 2011.
12. H. Chen. Sentiment and affect analysis of Dark Web forums: Measuring radicalization on the internet. In Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on, 2008.
13. Coling 2002. Conference on Computational Natural Language Learning (CoNLL-2002). Available at: `http://www.cnts.ua.ac.be/conll2002/`, Last visited March 2012.

14. W. Daelemans and H. Strik. Het Nederlands in taal- en spraaktechnologie: prioriteiten voor basisvoorzieningen. Technical report, Nederlandse Taalunie, 2002.
15. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. The VLDB Journal—The International Journal on Very Large Data Bases, 2007.
16. A. de Keijzer, M. van Keulen, and Y. Li. Taming Data Explosion in Probabilistic Information Integration. In Pre-Proceedings of the International Workshop on Inconsistency and Incompleteness in Databases (IIDB 2006), Munich, Germany, 2006.
17. B. Desmet and V. Hoste. Dutch Named Entity Recognition using Classifier Ensembles. Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands, 2010.
18. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In ACM SIGMOD Record, 2001.
19. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykois. Duplicate Record Detection: A Survey. IEEE Transactions on knowledge and data engineering, 2007.
20. J. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 2005.
21. D. Freitag. Machine Learning for Information Extraction in Informal Domains. PhD thesis, University of Washington, 1998.
22. N. Fuhr. A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In Proceedings of the 16th VLDB Conference Brisbane, Australia 1990, 1990.
23. GeoNames. GeoNames. Available at: `http://www.geonames.org/`, Last visited November 2011.
24. Google. Gmail. Available at: `http://gmail.com`, Last visited November 2011.
25. R. Goulart, V. Strube de Lima, and C. Xavier. A systematic review of named entity recognition in biomedical texts. Journal of the Brazilian Computer Society, March 2011.
26. R. Gupta and S. Sarawagi. Creating probabilistic databases from information extraction models. In Proceedings of the VLDB Endowment, 2006.
27. M. B. Habib and M. van Keulen. Information Extraction, Data Integration, and Uncertain Data Management: The State of The Art. Technical report, University of Twente, 2011.
28. P. Helland. If You Have Too Much Data, then "Good Enough" Is Good Enough. Communications of the ACM, 54(6), 2011.
29. J. Hoffart, M. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust Disambiguation of Named Entities in Text. In Proceedings of EMNLP, 2011.
30. J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: A Probabilistic Database Management System. In Proceedings of the 35th SIGMOD international conference on Management of data, 2009.

31. T. Huang, C. Dagli, S. Rajaram, E. Chang, M. Mandel, G. Poliner, and D. Ellis. Active Learning for Interactive Multimedia Retrieval. Proceedings of the IEEE, 2008.
32. R. Jampani, F. Xu, M. Wu, L. Perez, C. Jermaine, and P. Haas. MCDB: A Monte Carlo Approach to Managing Uncertain Data. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008.
33. Jasper Kuperus. Github: Probabilistic NER. Available at: `https://github.com/jasperkuperus/PNER`, Last visited April 2012.
34. D. Jurafsky and J. H. Martin. Speech And Language Processing. Pearson Education, 2009.
35. J. Kalbfleisch. Probability and Statistical Inference, Volume 1: Probability. Springer, 1985.
36. R. Klinger and K. Tomanek. Classical Probabilistic Models and Conditional Random Fields. Citeseer, 2007.
37. C. Koch, D. Olteanu, L. Antova, and J. Huang. MayBMS: A Probabilistic Database System User Manual, 2009.
38. C. Ku, A. Iriberri, and G. Leroy. Natural Language Processing and e-Government: Crime Information Extraction from Heterogeneous Data Sources. In The Proceedings of the 9th Annual International Digital Government Research Conference, 2008.
39. M. Lenzerini. Data integration: A theoretical perspective. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2002.
40. Machine Learning Group at University of Waikato. Weka. Available at: `http://www.cs.waikato.ac.nz/ml/weka/`, Last visited November 2011.
41. M. Magnani and D. Montesi. A Survey on Uncertainty Management in Data Integration. Journal of Data and Information Quality (JDIQ), 2010.
42. J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In Broadcast News Workshop'99 Proceedings, 1999.
43. A. Mansouri, L. S. Affendey, and A. Mamat. Named Entity Recognition Approaches. IJCSNS International Journal of Computer Science and Network Security,, 8(2), 2008.
44. Max Planck Institute for Informatics. AIDA Web Interface. Available at: `https://d5gate.ag5.mpi-sb.mpg.de/webaida/`, Last visited November 2011.
45. MayBMS. MayBMS - A Probabilistic Database Management System. Available at: `http://maybms.sourceforge.net/`, Last visited September 2011.
46. E. Minkov, R. Wang, and W. Cohen. Extracting Personal Names from Email: Applying Named Entity Recognition to Informal Text. In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, 2005.
47. D. Nadeau and S. Sekine. A survey of named entity recognition and classification. Lingvisticae Investigationes, 30(1), 2007.

48. Nederlandse Taalunie. STEVIN: Spraak- en Taaltechnologische Essentiële Voorzieningen In het Nederlands. Available at: http://taalunieversum.org/taal/technologie/stevin/, Last visited October 2011.
49. Netherlands Forensic Institute. Kecida: Kennis- en expertisecentrum voor intelligente data-anlyse. Available at: http://www.forensischinstituut.nl/dienstverlening/producten/kecida.aspx, Last visited November 2011.
50. A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of LREC 2010, 2010.
51. PostgreSQL Global Development Group. Postgresql. Available at: http://www.postgresql.org/, Last visited October 2011.
52. L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77(2), 1989.
53. M. Reynaert, N. Oostdijk, O. De Clercq, H. Heuvel, and F. Jong. Balancing SoNaR: IPR versus processing issues in a 500-million-word written Dutch reference corpus. In Proceedings of the Seventh Conference of International Language Resources and Evaluation (LREC'10), 2010.
54. S. Sarawagi. Information Extraction. Foundations and Trends in Databases, 1(3), 2008.
55. A. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. IEEE Computer Society, 2006.
56. I. Schuurman, V. Hoste, and P. Monachesi. Interacting Semantic Layers of Annotation in SoNaR, a Reference Corpus of Contemporary Written Dutch. In Proceedings of the Seventh International Conference on Linguistic Resources and Evaluation (LREC-2010), Valletta, Malta, 2010.
57. S. Sekine and C. Nobata. Definition, dictionaries and tagger for Extended Named Entity Hierarchy. In Proceedings of the Language Resources and Evaluation Conference (LREC), 2004.
58. P. Sen, A. Deshpande, and L. Getoor. Exploiting shared correlations in probabilistic databases. Proceedings of the VLDB Endowment, 1(1), 2008.
59. Stanford Trio Project. Trio: A system for integrated management of data, uncertainty and lineage. Available at: http://infolab.stanford.edu/trio/, Last visited October 2011.
60. Stanford University - Natural Language Processing. Stanford Named Entity Recognizer (NER). Available at: http://nlp.stanford.edu/software/CRF-NER.shtml, Last visited March 2012.
61. D. Suciu. Mystiq. Available at: http://www.cs.washington.edu/homes/suciu/project-mystiq.html, Last visited October 2011.
62. J. Turmo, A. Ageno, and N. Catala. Adaptive Information Extraction. ACM Computing Surveys (CSUR), 38(2), July 2006.
63. Twitter. Twitter. Available at: https://twitter.com/, Last visited November 2011.

64. M. van Keulen. Onzekere databases. DB/M : database magazine, 2010.
65. M. van Keulen and A. de Keijzer. Qualitative Effects of Knowledge Rules and User Feedback in Probabilistic Data Integration. The VLDB Journal, 2009.
66. M. van Keulen and M. B. Habib. Handling uncertainty in information extraction. In Proceedings of the 7th International Workshop on Uncertainty Reasoning for the Semantic Web, Bonn, Germany, 2011.
67. M. van Keulen and M. B. Habib. Named Entity Extraction and Disambiguation: The Reinforcement Effect. In Proceedings of the 5th International Workshop on Management of Uncertain Data, MUD 2011, Seatle, USA, 2011.
68. D. Volkskrant. Weer arrestatie na dreigtweet. Available at: `http://www.volkskrant.nl/vk/nl/2686/Binnenland/article/detail/1875812/2011/04/15/Weer-arrestatie-na-dreigtweet.dhtml`, Last visited November 2011.
69. T. Vosse. Detecting and Correcting Morpho-syntactic Errors in Real Texts. In Proceedings of the third conference on Applied natural language processing, 1992.
70. N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of Proper Names in Text. In Proceedings of the fifth conference on Applied natural language processing, 1997.
71. H. W. Wallach. Conditional Random Fields: An Introduction. Rapport technique MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, 50, 2004.
72. M. Wick. Representing Uncertainty in Databases with Scalable Factor Graphs. In In Proceedings of the 32nd international conference on very large data bases (VLDB 2010), 2010.
73. C. Will. Comparing human and machine performance for natural language information extraction: results for English microelectronics from the MUC-5 evaluation. In Proceedings of the 5th conference on Message understanding, 1993.
74. M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum. AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. In Proceedings of the VLDB Endowment, 2011.
75. X. Zhu. CS838-1 Advanced NLP: Conditional Random Fields. Technical report, The University of Wisconsin Madison, 2007.

Acronyms

This chapter provides a list of acronyms used in this thesis. Where some of these acronyms require further explanation, the Glossary provides it.

CRF Conditional Random Field

DBAL Database Abstraction Layer

HMM Hidden Markov Model

IE Information Extraction

Kecida Knowledge- and Expertise Centre for Intelligent Data Analysis

KEES Kecida Entity Extraction Software

MEM Maximum Entropy Model

NEE Named Entity Extraction

NED Named Entity Disambiguation

NEN Named Entity Normalisation

NER Named Entity Recognition

NERC Named Entity Recognition and Classification

NFI Netherlands Forensic Institute

NLP Natural Language Processing

OCR Optical Character Recognition

PNER Probabilistic Named Entity Recognition

POS Part of Speech

RWO Real World Object

SoNaR STEVIN Nederlandstalig Referentiecorpus

SQL Structured Query Language

STEVIN Spraak- en Taaltechnologische Essentiële Voorzieningen in het Nederlands

SVM Support Vector Machine

WST World Set Table

Glossary

This chapter provides explanation on some regular used terms in this document, important for a better understanding.

Annotation The assignment of a type to an *Entity*. Entities can have multiple *Annotation*s scored with probabilities.

Boundary The position of an entity in the text, defined by its start and end character position within the document.

Entity Representation of a *Named Entity*, defined by its text and boundary.

Entity Bundle A collection of *Entities* that span the same boundary or are part of a bigger boundary, representing structural ambiguity.

Entity Disambiguation The process of removing semantic, structural or reference ambiguity given an ambiguous entity.

Label Same as *Annotation*.

Named Entity Anything that can be referred to using a proper name. Common proper names subjected to extraction are names of persons and organisations, locations, phone numbers, etc. [34]

Named Entity Recognition The process of identification, classification and extraction of named entities from unstructured texts.

Probabilistic Named Entity Recognition A probabilistic approach to Named Entity Recognition that avoids extraction errors by not making decisions at extraction time, but keeping all alternative answers scored with a probability.

Appendix A

PNER Framework Implementation

This appendix elaborates on the implementation details, describing the data model and the Probabilistic Named Entity Recognition (PNER) framework into more detail on implementation level.

A.1 Data Model

Figure A.1 shows the data model which is used in the implementation of the PNER framework. This section provides a quick description of the objects.

Document represents the input. All input is converted into a `Document` of a certain `type` (e.g. file, Twitter) and optional `subtype`. The `source` shows where it comes from, in case of a file, the path. Finally, the `contents` are the actual contents of this document.

Metadata represents a key-value pair carrying metadata for a document. This can be anything the user sees as interesting, e.g. a last modification date of a document.

Entity represents a possible entity found in a document. The `text` is the actual content of this entity, found on the location in the document described by the `boundary`. When this entity is part of a set of structural ambiguous entities, `structural` is true and `parentBoundary` is the boundary containing all these entities (see A.2.1.5). Finally, `line` denotes on which line of the document this entity appears.

Boundary represents a location within a document, using a `start` and `end` index and supplying some methods for detecting structural ambiguity.

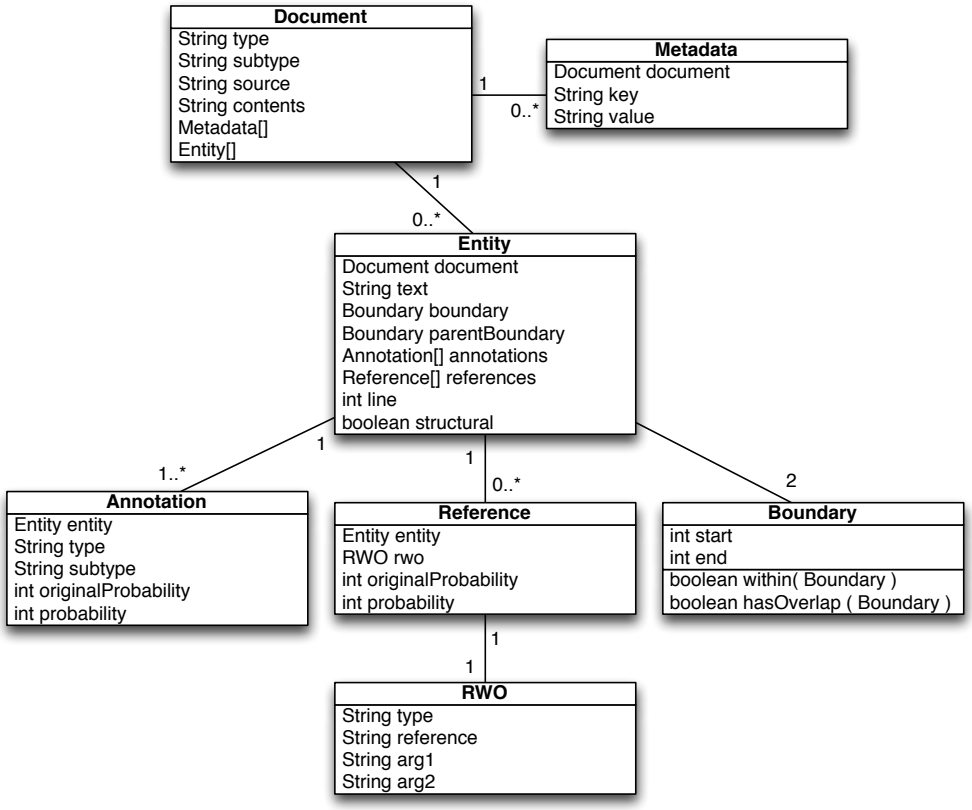

Fig. A.1. The Data Model for the PNER Framework

Annotation represents a possible classification of the entity of type `type` and with an optional `subtype`. The `originalProbability` is the original probability which will not be modified. The `probability` is the working value and will have the same start value as `originalProbability`.

Reference represents a possible reference to a real world object (RWO), representing Reference Ambiguity. The probabilities are analogous to `Annotation`.

RWO represents a Real World Object. The `type` is what sort of RWO it refers to, this can e.g. be a GeoNames [23] reference. The `reference` is the actual reference, which can e.g. be the ID of the entry in GeoNames. Furthermore, `arg1` and `arg2` can be used when more parameters are needed for unique identification.

A.2 Framework

Figures 4.1 and 4.2 provide an abstract view of the process for Probabilistic NER. Figure A.2 shows the framework which has been derived from these processes and is implemented during this project and used during experimentation.

Due to the proportion of the whole problem and the fact that not all problems are within the scope of this research project, concessions had to be made. Therefore, some parts, like the probabilistic database, are implemented using a more pragmatic approach. Due to this, there is an increased chance that in the future some parts of the framework will be replaced with other implementations. Therefore, the focus during implementation was on keeping the framework as abstract as possible. By keeping the framework very abstract, a new NER approach or new probabilistic database can easily be built within the process by simply replacing only that component and satisfying its interface.

A.2.1 Extraction Process

The part above the Database Abstraction Layer in Figure A.2 depicts the extraction process of the PNER framework. This section elaborates on the implementation details of the extraction process.

A.2.1.1 Reading

The framework allows multiple ways of reading data as input for the extraction process. By default a filesystem reader is implemented which can convert files or directories into documents. Possible extensions here are readers which read Twitter messages, allow streaming data, etc.

A.2.1.2 Extraction

The dimension of the extraction phase depends on the underlying NER method. When a rule based approach that does not supply in probabilities is used for NER, all sub-phases should be implemented. Is the underlying NER method e.g. a machine learning approach which provides probabilities, both Probability Calculation and Normalisation can be omitted.

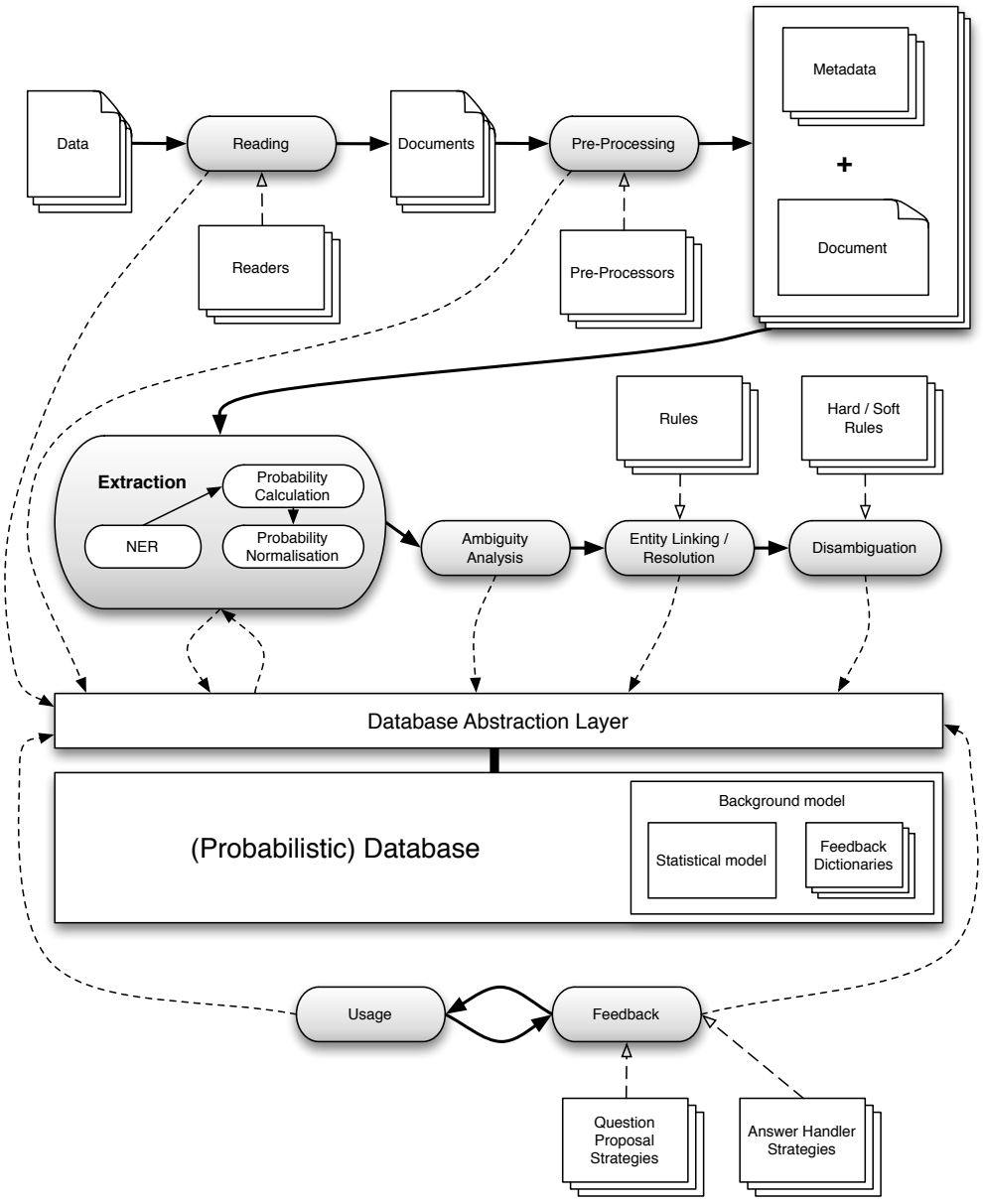

Fig. A.2. The Probabilistic NER framework

The NER phase is a wrapper for the actual NER process. In this phase, the desirable NER method should be linked to the framework, making it accept

documents containing metadata and making sure the phase delivers a set of entities complying to the data model described in section A.1. As an example, the NER method of Kecida, KEES, only accepts filesystem input and persists the extraction results in an SQL Server database as so far. Creating a wrapper for KEES incorporates the preparation of data to the filesystem, executing KEES and then extraction of the results from the SQL Server.

Probability Distribution Both the Probability Calculation and Normalisation are part of a *Probability Manager* which is given to the NER Extractor to distribute the probabilities. A Probability Calculator is used to assign probabilities to annotations of an entity.

When the NER method uses multiple detectors to find the entities (e.g. by using regular expressions in combination with dictionaries) and the Probability Calculator works the same way, the Probability Normaliser can be used to normalize the probabilities so they add up to a maximal of 100% per entity.

A.2.1.3 Ambiguity Analysis

In the Ambiguity Analysis phase, a flat list of entities which is the result of the extraction phase is analyzed for ambiguity. Both semantic and structural ambiguity are detected following the algorithm described in Listing A.1

```
1  // List of all processed entities
2  Map <Boundary, Entity> entities;
3  // Fast lookup for boundaries with structural ambiguity
4  Map <Boundary, List<Entity>> parentMap;
5
6  // Loop over every entity to check for ambiguity
7  for( Entity entity : input ) {
8      if( entities.contains( entity.getBoundary() ) ) {
9          // Exact boundary exists -> Semantic Ambiguity
10         entities.get( entity.getBoundary() )
11             .getAnnotations().addAll(
12                 entity.getAnnotations() );
13     } else {
14         // Start values for merged results
15         List<Entity> merge;
16         Boundary mergeBoundary = entity.getBoundary();
17         merge.add( entity );
```

```
18
19          // Find overlapping entities, merge them
20          for( Map.Entry entry : parentMap.entrySet() ) {
21              // Fast lookup values
22              Boundary parent = entry.getKey();
23              List<Entity> children = entry.getValue();
24
25              if( entity.getBoundary().within( parent ) ) {
26                  // The entity lies within these entities
27                  entity.setParentBoundary( parent );
28                  children.add( entity );
29
30                  // Update boundaries and ambiguity
31                  children.setParentBoundary( parent );
32                  children.setStructuralAmbiguity( true );
33              } else if( entity.getBoundary()
34                      .hasOverlap( parent ) ) {
35                  // Add children for the merge operation
36                  merge.addAll( children );
37
38                  // Set a new merged (extended) boundary
39                  mergeBoundary = new Boundary(
40                      min( parent.start, entity.start ),
41                      max( parent.end, entity.end ) );
42
43                  // Remove this entry from the map, we'll
44                  // add it later, merged
45                  parentMap.remove( parent );
46              }
47          }
48
49          // Add this entity to the entity lookup
50          entities.put( entity.getBoundary(), entity );
51
52          // Update boundaries and ambiguity
53          if( merge.size() > 1 ) {
54              merge.setParentBoundary( mergeBoundary );
```

```
55                merge.setStructuralAmbiguity( true );
56                parentMap.put( mergeBoundary, merge );
57            }
58        }
59    }
```

Listing A.1. Ambiguity Analysis Algorithm in Pseudo code

A.2.1.4 Rule Engines

The phases *Pre-Processing*, *Entity Linking / Resolution* and *Disambiguation* show quite some common ground in the fact that they can differ for certain situations and the fact that the user should be able to extend these phases in order to add functionality. In this sense, these phases can all be considered as a *Rule Engine*, where the user can define, activate or deactivate rules to control the functionality which is applied in these phases.

For the Pre-Processing phase, a rule corresponds to a Pre-Processor which can Pre-Process a certain type of document. For the Entity Linking / Resolution phase, rules can be defined for defining probabilistic links between entities and Real World Objects. Finally, in the Disambiguation step, rules can be defined for disambiguating certain results of which it is absolutely certain they are no entities or that it is most likely that they are no entities. Distinguishing between hard and soft rules makes it possible to create rules which delete entities and rules which adjust probabilities.

A.2.1.5 Probabilistic Database

Due to limitations of current probabilistic database models and the fact that building a 100% correct probabilistic model would take too much time and not make it possible to lay the focus on PNER and Targeted Feedback, there is chosen to create a more pragmatic implementation of a probabilistic database on top of MySQL, supplying enough features to operate. More information on the actual probabilistic model can be found in section 4.3 and appendix D.

Due to this pragmatic approach, it is quite likely that in the future the database implementation is to be replaced. Therefore, the *Database Abstraction Layer* is introduced. Every communication with the database goes through this layer. Therefore, implementing a new database would only imply building a new Abstraction Layer for that database.

The background model is a part within the probabilistic database which holds information regarding the answers given by the user out of the Targeted

Feedback. The base of the background model is a log of all activity produced by user feedback. Other parts like a statistical model are strategy dependent.

A.2.2 Usage and Feedback

In this phase, the user browses through the data using the developed query language, described in appendix B. When a feedback strategy is activated while browsing through the data, the strategy will pose a question according the query results which the user can then decide to answer to increase the data quality. Figures A.3, A.4 and A.5 show screenshots of the browser, providing an impression of the usage and feedback phases.

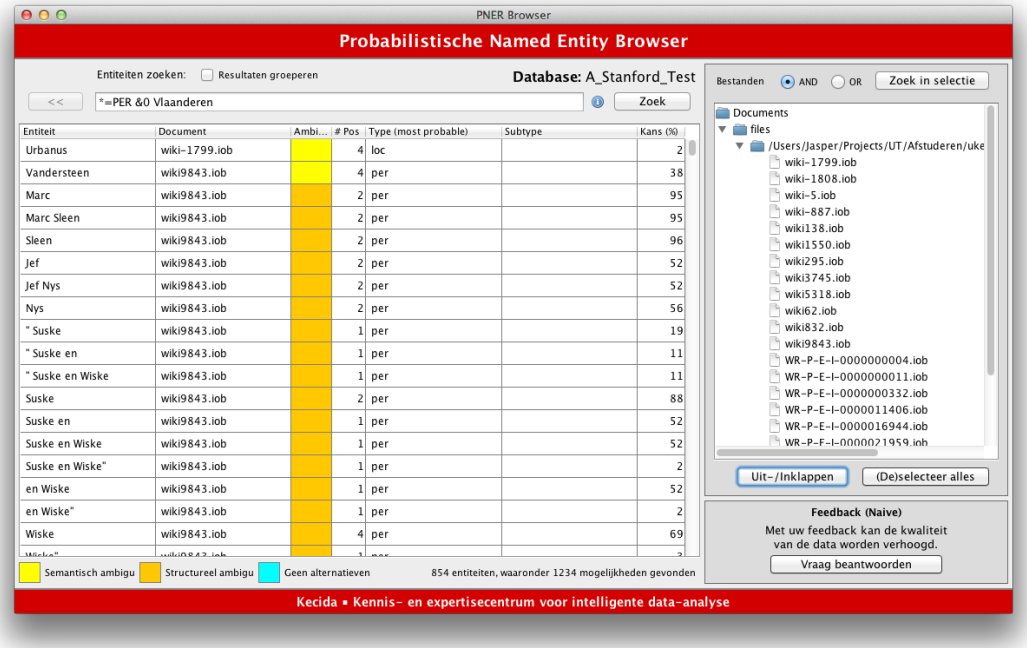

Fig. A.3. The PNER Browser: Browser

Fig. A.4. The PNER Browser: Single Document View

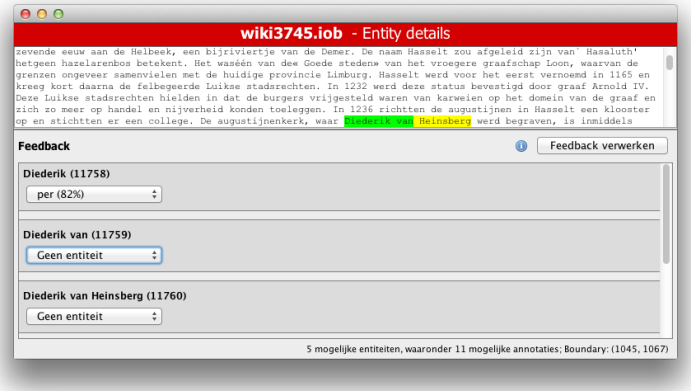

Fig. A.5. The PNER Browser: Resolving Ambiguity, Targeted Feedback

Appendix B

PNER Query Language

This appendix provides a quick guide on the PNER Query Language designed which can be used in the PNER Browser to query the PNER results.

B.1 Simple Queries

Simply supplying one string as query, the Query Interpreter will look through the text of all the entities and return everything that matches that string. By using the wildcards as shown in table B.1, the texts do not have to match the exact text anymore.

Wildcards	
*	Matches one or more characters
?	Matches exactly one character

Table B.1. PNER Query Language - Wildcards

B.2 Type Constraints

Type constraints can be added by incorporating an equals sign in the query, e.g. `*=PER` will look for everything that has as possible type `PER`. For the type

information, the Query Interpreter will look at both the `type` and the `subtype` of an annotation. Also in type constraints the wildcards presented in table B.1 can be used. As example, when incorporating data from multiple extractors of which the one uses `PER` and the other `person`, the query `*=PER*` will return annotations complying with either `PER` or `person`.

B.3 Relational Queries

In the current version of the PNER Query Language, a relation is defined as two entities that are within `x` number of lines of each other. Within the query, the variable `x` can be defined, taking control on how far the relation reaches. The ampersand (`&`) is used as relational operator, directly followed by the number of lines, e.g. `&1`. Table B.2 provides some examples of relational queries.

Query	**Explanation**
***=PER &0 Amsterdam**	All persons within the same line as 'Amsterdam' entities
***=PER &1 Amsterdam=LOC**	All persons within max. 1 line distance of an 'Amsterdam' entity which is a location
Kees &2 *=LOC	All 'Kees' entities within max. 2 line distance of an entity of type location

Table B.2. PNER Query Language - Relational Query Examples

B.4 Document Constraints

Although the PNER Query Language can not express document constraints, the PNER Browser allows adding such constraints to the query. Using these constraints, all of the above queries can be executed on a set of documents using an *AND* or *OR* operation, respectively selecting entities that appear in all of the selected documents and selecting entities that appear in either of the selected documents. Without any document constraints the Query Interpreter searches in the whole resultset.

Appendix C

Stanford NER CRF Model Training

This research project uses Stanford NER's CRF Classifier [60,20] as underlying NER approach. This Appendix elaborates on how SoNaR is partitioned into multiple datasets, including the training dataset, and on how the CRF model is trained.

C.1 Input: SoNaR Partitioning

As section 6.1.1 described, the wikipedia text type documents are partitioned in three parts, a *training partition*, *development partition* and a *validation partition*. Table C.1 gives a detailed view of which documents are in which partition.

When partitioning the texts, the aim was to have three balanced partitions containing texts in the whole range of small (∼100 words) and large (∼8000 words) texts. First, texts of matching sizes among the whole range were assigned to each of the three sets to have an equally balanced base. When having one of the sets sized at ∼75000 words, the other two sets were raised to an equal size, leaving lots of smaller documents unused. These smaller documents were then assigned to the training set to have a bigger dataset to train on.

C.2 CRF Training Parameters

The Stanford NER Classifier uses an approach based on Conditional Random Fields (CRFs). Table C.1 provides a list of documents that are used to train this

Training	Development	Validation
WR-P-E-I-0000000001, WR-P-E-I-0000000008,	WR-P-E-I-0000000004,	WR-P-E-I-0000000006,
WR-P-E-I-0000000045, WR-P-E-I-0000004258,	WR-P-E-I-0000000011,	WR-P-E-I-0000000014,
WR-P-E-I-0000006366, WR-P-E-I-0000015007,	WR-P-E-I-0000000332,	WR-P-E-I-0000004745,
WR-P-E-I-0000020972, WR-P-E-I-0000024561,	WR-P-E-I-0000011406,	WR-P-E-I-0000013937,
WR-P-E-I-0000036627, WR-P-E-I-0000039081,	WR-P-E-I-0000016944,	WR-P-E-I-0000019936,
WR-P-E-I-0000039589, WR-P-E-I-0000041235,	WR-P-E-I-0000021959,	WR-P-E-I-0000022743,
WR-P-E-I-0000041531, WR-P-E-I-0000044854,	WR-P-E-I-0000026563,	WR-P-E-I-0000027197,
WR-P-E-I-0000050394, WR-P-E-I-0000108667,	WR-P-E-I-0000027216,	WR-P-E-I-0000032165,
wiki-60, wiki-90, wiki-135, wiki-204, wiki-209,	WR-P-E-I-0000039352,	WR-P-E-I-0000040286,
wiki-334, wiki-342, wiki-349, wiki-356, wiki-493,	WR-P-E-I-0000045368,	WR-P-E-I-0000042791,
wiki-572, wiki-659, wiki-702, wiki-737, wiki-859,	WR-P-E-I-0000086197,	WR-P-E-I-0000050211,
wiki-888, wiki-935, wiki-1112, wiki-1181,	wiki-5, wiki-62,	wiki-11, wiki-89,
wiki-1250, wiki-1327, wiki-1330, wiki-1532,	wiki-138, wiki-295,	wiki-384, wiki-415,
wiki-1553, wiki-1566, wiki-1609, wiki-1617,	wiki-832, wiki-887,	wiki-866, wiki-889,
wiki-1625, wiki-1635, wiki-1652, wiki-1820,	wiki-1550, wiki-1799,	wiki-1058, wiki-1094,
wiki-1846, wiki-1928, wiki-1941, wiki-3668,	wiki-1808, wiki-3745,	wiki-1400, wiki-1541,
wiki-3747, wiki-3781, wiki-3821, wiki-3822,	wiki-5318, wiki-9843,	wiki-1823, wiki-3828,
wiki-3823, wiki-3824, wiki-3825, wiki-3826,		wiki-5593, wiki-6879,
wiki-3866, wiki-4868, wiki-4941, wiki-4979,		wiki-6983, wiki-7064,
wiki-5090, wiki-5107, wiki-5176, wiki-5177,		wiki-7355, wiki-7891,
wiki-5452, wiki-5594, wiki-5741, wiki-6236,		wiki-8628, wiki-8894,
wiki-6253, wiki-6532, wiki-6616, wiki-6880,		wiki-9844
wiki-6930, wiki-6984, wiki-7097, wiki-7218,		
wiki-7298, wiki-7542, wiki-7543, wiki-7585,		
wiki-7793, wiki-8030, wiki-8238, wiki-8783,		
wiki-9515, wiki-9720, wiki-9809		

Table C.1. Partitioning of SoNaR wikipedia texts

CRF model on. SoNaR provides 6 different annotations, which are summarized in Table C.2.

SoNaR Annotation Types	
PER	Person
LOC	Location
ORG	Organisation
EVE	Event
PRO	Product
MISC	Miscellaneous

Table C.2. SoNaR Annotation Types

Because the aim of this research project was not to create the best tuned NER solution, but was on a Probabilistic approach to NER, Stanford NER is merely used as starting point, representing results containing ambiguity and imperfect probabilities. Therefore, the way in which the Stanford NER model is trained is not tuned indefinitely and is closely related to the example training properties provided by Stanford. Table C.3 shows the features on which the CRF model is trained.

Feature	Value	Description
maxLeft	*1*	The number of tokens to the left to be cached to run the Viterbi algorithm
useWord	*true*	Provides features for the word
useClassFeatures	*true*	Include a feature for the class
useNGrams	*true*	Make features from letter n-grams
noMidNGrams	*true*	Do not include n-grams that contain neither the beginning or the end of the word
maxNGramLeng	*6*	Exclude n-grams above this size
usePrev	*true*	Enables previous features
useNext	*true*	Enables next features
useDisjunctive	*true*	Include giving disjunctions of words
useSequences	*true*	Enable sequences
usePrevSequences	*true*	Enable previous sequences
useTypeSeqs	*true*	Enable basic zero order word shape features
useTypeSeqs2	*true*	Enable additional first and second order word shape features
useTypeySequences	*true*	Enable first order word shape patterns
wordShape	*chris2useLC*	Enable additional word shape features

Table C.3. Stanford CRF Training Features

Appendix D

Random Variables Approach

This appendix provides a more detailed explanation of the Random Variables approach to the probabilistic model and provides an extended example.

D.1 Random Variables

A complete approach to a probabilistic model for NER is the random variables approach. This research project takes both semantic ambiguity and structural ambiguity into account. The probability of an entity being of a certain type relies on both of these ambiguity types, the probability of that entity being of that type and the probability that this entity exists with these boundaries.

The random variables will then represent the probabilities for an entity bundle. Both the type of the individual entities and the structure of the bundle can be expressed in random variables. The first random variable, say x_1, will represent the permutations of the types of the entities within the bundle. The second variable, say x_2, will represent the permutations of the structure of the entities. All possible values of both x_1 and x_2 are then inserted in the *World Set Table*, where they are provided with a probability.

As with the current approach, a probability for the structure is not provided by Stanford NER. Therefore, this probability still has to be estimated in a way as has been presented in section 4.3. The probabilities for the types can be calculated with the (conditional) probabilities provided by Stanford NER, in analogy to section 4.3.

D.2 Example

This section describes an extended example of the probabilistic model using random variables and how this theory is translated into a database model.

Among the extracted entities in the development partition is the entity bundle for the text `European Centre Brussels`. Table D.1 shows the, for this example, simplified possible entities and annotations for the entity bundle. For simplicity, probabilities are omitted, how they can be calculated is already described in section 4.3.

	Entities					
	European	*Centre*	*Brussels*	*European Centre*	*Centre Brussels*	*European Centre Brussels*
Annotations	ORG LOC	ORG LOC	ORG LOC	ORG LOC	ORG LOC	ORG

Table D.1. Random Variables: Extracted Entity Possibilities

In the database, entities and the possible annotations are split into two tables. Table D.1 can easily be translated into Table D.2.

Entities

id	*Entity*	...
E	European	...
C	Centre	...
B	Brussels	...
EC	European Centre	...
CB	Centre Brussels	...
ECB	European Centre Brussels	...

Table D.2. Random Variables: Entities Table

Before going into details on the annotations table, table D.3 shows the *World Set Table*, storing the random variables and their probabilities (which are omitted in this example).

World Set Table

Var	*Value*	*Human Readable*
x_1 **: Annotation Type**		
x_1	0	E=ORG ∧ C=ORG ∧ B=ORG
x_1	1	E=LOC ∧ C=ORG ∧ B=ORG
x_1	2	E=ORG ∧ C=LOC ∧ B=ORG
x_1	3	E=ORG ∧ C=ORG ∧ B=LOC
x_1	4	E=LOC ∧ C=LOC ∧ B=ORG
x_1	5	E=ORG ∧ C=LOC ∧ B=LOC
x_1	6	E=LOC ∧ C=ORG ∧ B=LOC
x_1	7	E=LOC ∧ C=LOC ∧ B=LOC
x_2 **: Structure**		
x_2	0	E ∧ C ∧ B
x_2	1	EC ∧ B
x_2	2	E ∧ CB
x_2	3	ECB

Table D.3. Random Variables: World Set Table

For every combination of x_1 and x_2 where an entity can be of a certain type, the annotations table stores an entry. Table D.4 shows the annotations table derived from the previous tables.

Before calculating the probability $P(E = ORG)$, all possible annotations with *Entity* `E` and *Type* `ORG` are to be retrieved. The summation of the product of the x_1, x_2 tuples corresponds to the probability of `E=ORG`.

Apart from the fact that this is a complete probabilistic model, it also represents mutual exclusiveness in a strong way. Choosing a certain value for a random variable directly eliminates other possible types or structures. E.g., $x_2 = 2$ di-

Annotations

Entity	*Type*	x_1	x_2
E	ORG	0	0
E	ORG	2	0
E	ORG	3	0
E	ORG	5	0
E	ORG	0	2
E	ORG	2	2
E	ORG	3	2
E	ORG	5	2
E	LOC	1	0
E	LOC	4	0
E	LOC	6	0
E	LOC	1	2
E	LOC	4	2
E	LOC	6	2
E	LOC	7	2
C	ORG	0	0

Entity	*Type*	x_1	x_2
C	ORG	1	0
C	ORG	3	0
C	ORG	6	0
C	LOC	2	0
C	LOC	4	0
C	LOC	5	0
C	LOC	7	0
B	ORG	0	0
B	ORG	1	0
B	ORG	2	0
B	ORG	4	0
B	ORG	0	1
B	ORG	1	1
B	ORG	2	1
B	ORG	4	1
B	LOC	3	0

Entity	*Type*	x_1	x_2
B	LOC	5	0
B	LOC	6	0
B	LOC	3	1
B	LOC	5	1
B	LOC	6	1
B	LOC	7	1
EC	ORG	0	1
EC	ORG	3	1
EC	LOC	4	1
EC	LOC	7	1
CB	ORG	0	2
CB	ORG	1	2
CB	LOC	5	2
CB	LOC	7	2
ECB	ORG	0	3

Table D.4. Random Variables: Annotations Table

rectly eliminates the structure `ECB` explicitly, whereas in the current pragmatic approach this is implicitly represented.

D.3 Reference Ambiguity

Although the above example has only shown how to cope with Structural and Semantic ambiguity, representing Reference ambiguity works quite similar. Another random variable, x_3, can be introduced to represent all possible references. Analogous to x_1 and x_2, x_3 can be incorporated in the *World Set Table* and the *Annotations* table.

Appendix E

Feedback Simulation Results

This appendix provides an overview of all graphs produced by experimenting with the multiple feedback strategies. The graphs are first ordered by Answer Handler Strategy, then by Question Proposal Strategy and then by query.

E.1 Basic Database Strategy

E.1.1 Naive Strategy

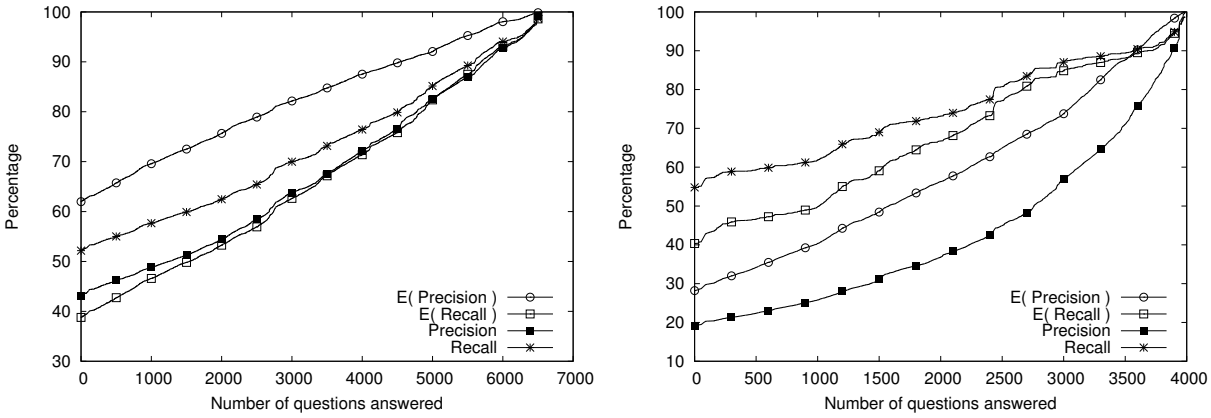

Fig. E.1. Basic DB - Naive; Left: A (*); Right: B (*=PER)

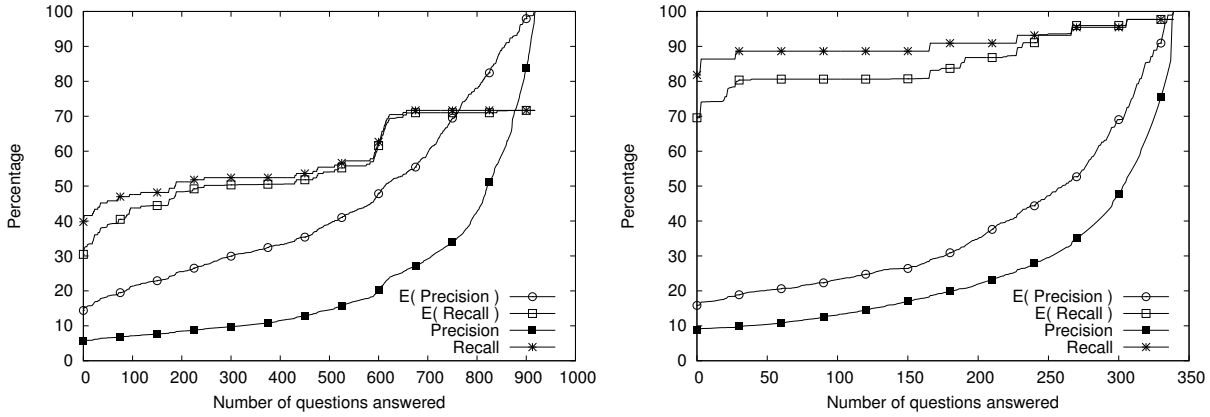

Fig. E.2. Basic DB - Naive; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

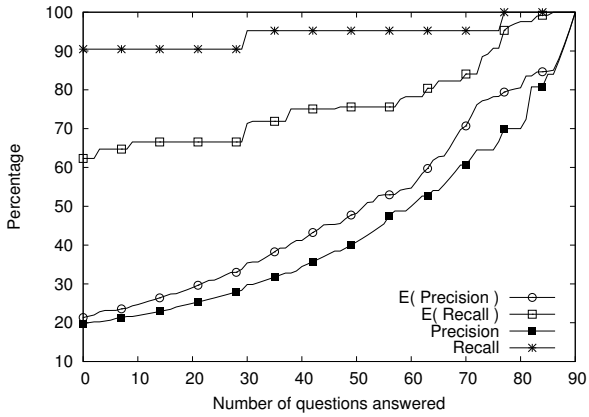

Fig. E.3. Basic DB - Naive; E (*=PER &1 Verenigde Staten)

E.1.2 Random Strategy

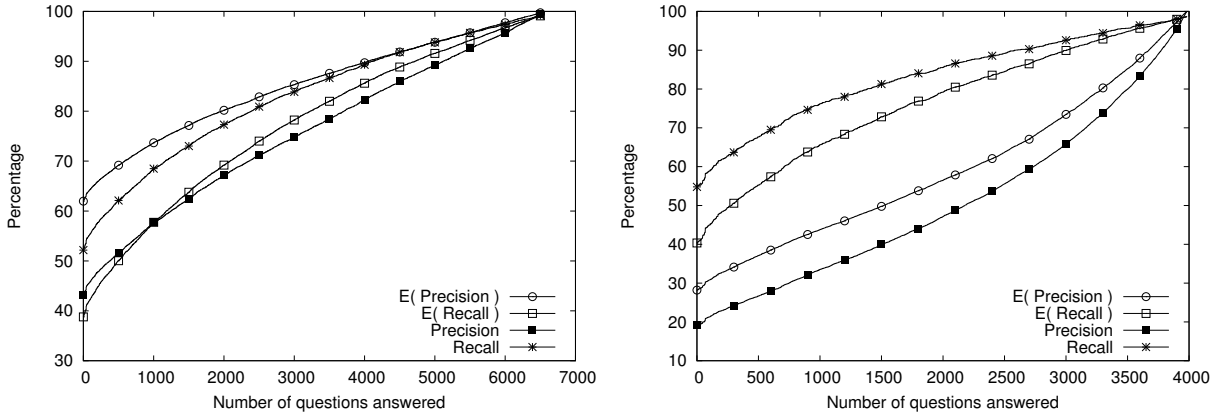

Fig. E.4. Basic DB - Random; Left: A (*); Right: B (*=PER)

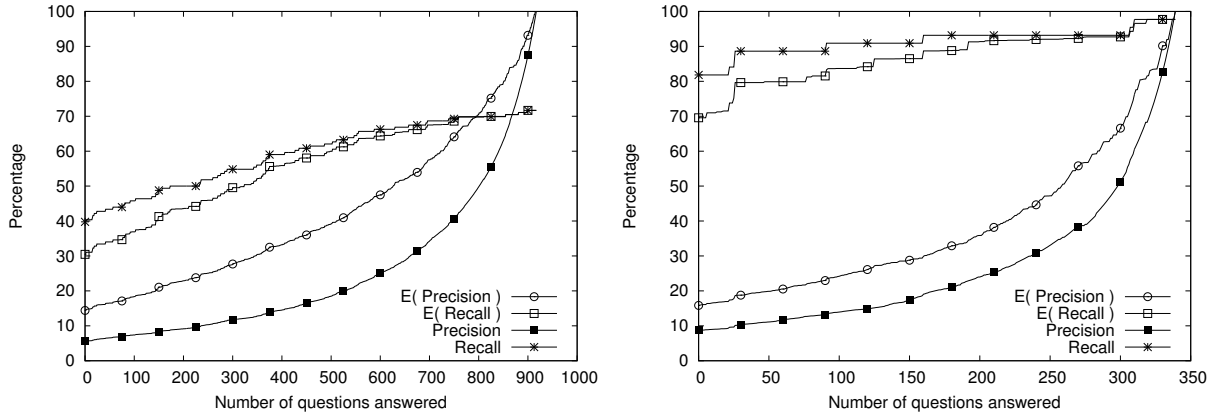

Fig. E.5. Basic DB - Random; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

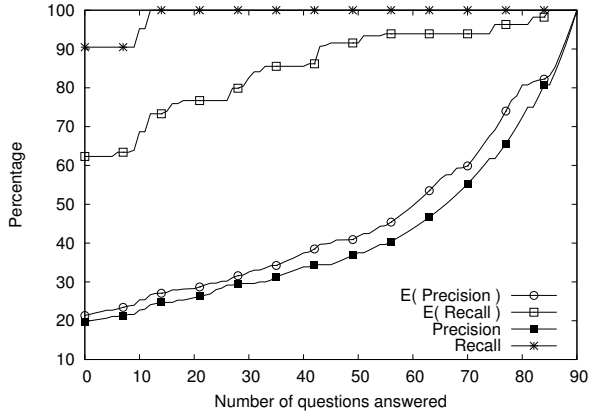

Fig. E.6. Basic DB - Random; E (*=PER &1 Verenigde Staten)

E.1.3 Structural Strategy

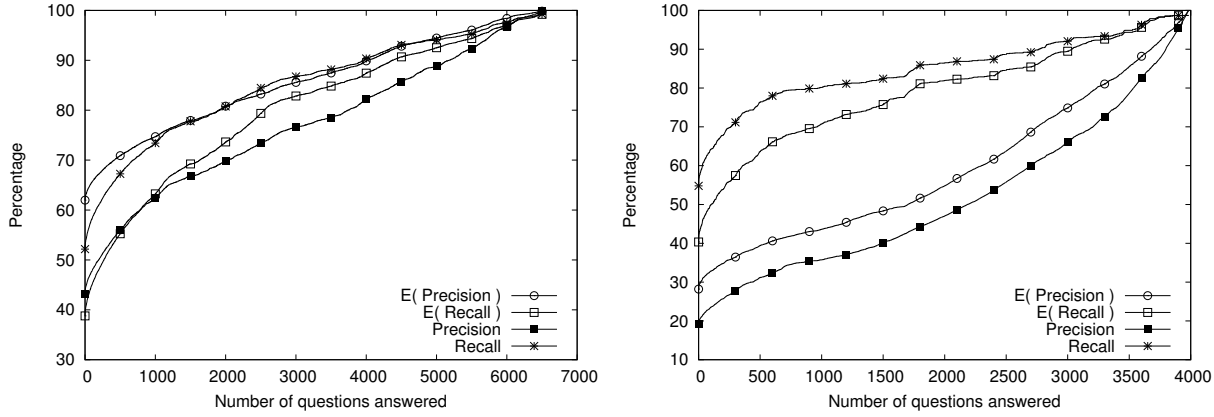

Fig. E.7. Basic DB - Structural; Left: A (*); Right: B (*=PER)

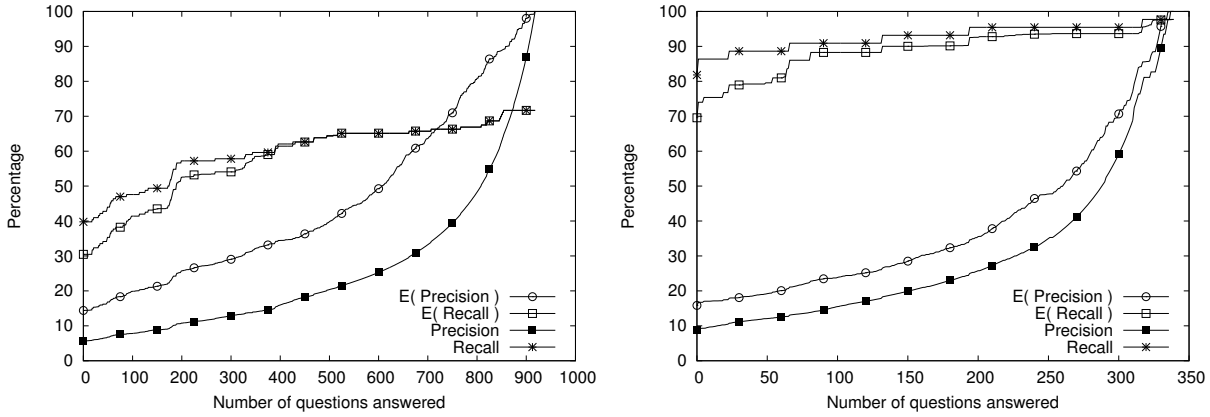

Fig. E.8. Basic DB - Structural; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

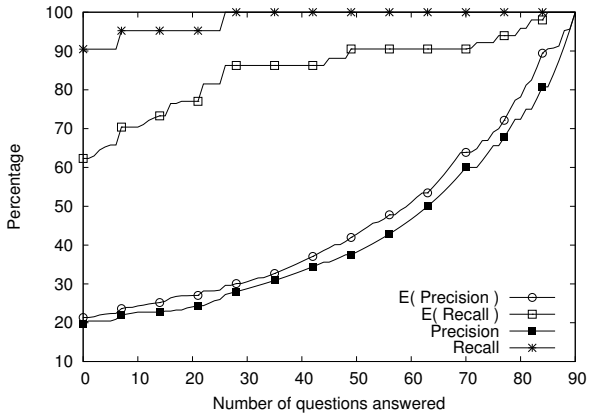

Fig. E.9. Basic DB - Structural; E (*=PER &1 Verenigde Staten)

E.1.4 Genius Strategy

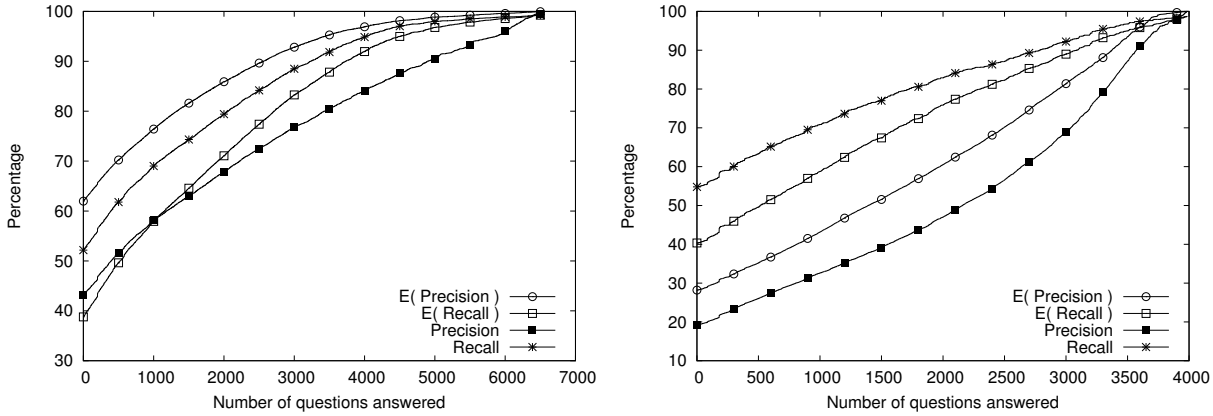

Fig. E.10. Basic DB - Genius; Left: A (*); Right: B (*=PER)

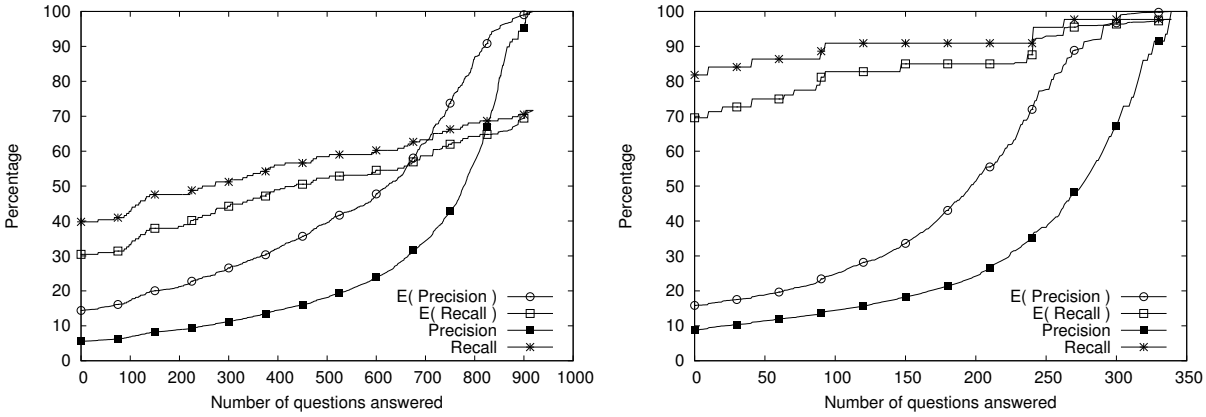

Fig. E.11. Basic DB - Genius; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

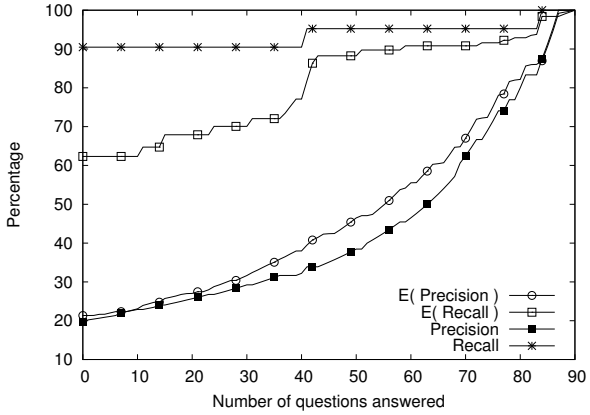

Fig. E.12. Basic DB - Genius; E (*=PER &1 Verenigde Staten)

E.1.5 Cluster Strategy

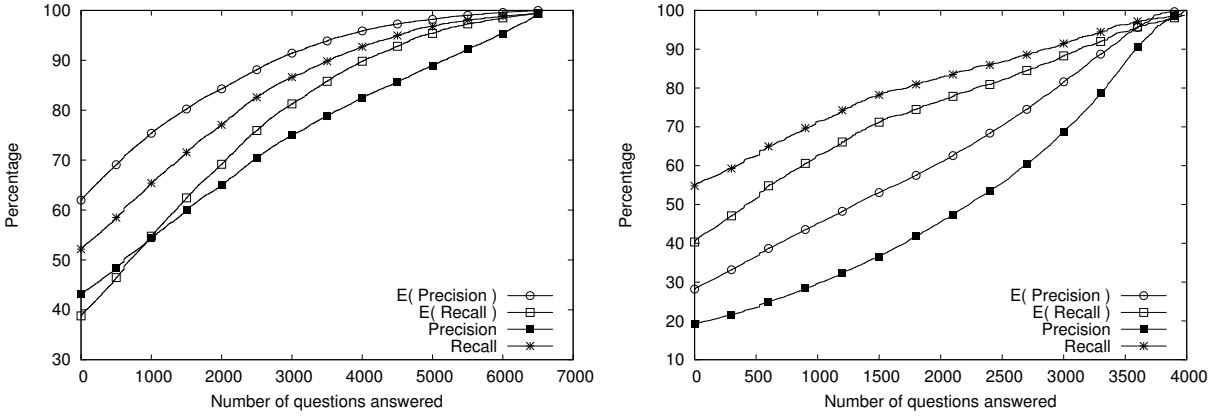

Fig. E.13. Basic DB - Cluster; Left: A (*); Right: B (*=PER)

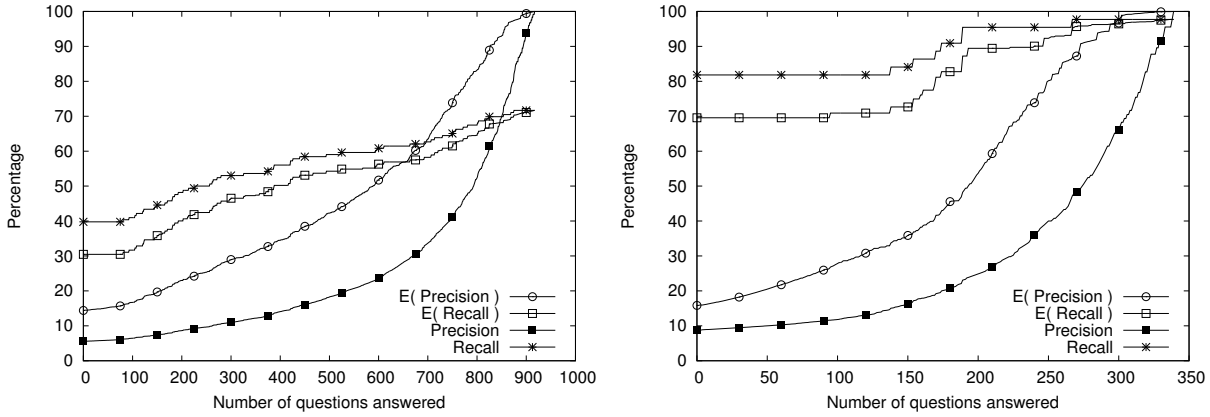

Fig. E.14. Basic DB - Cluster; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

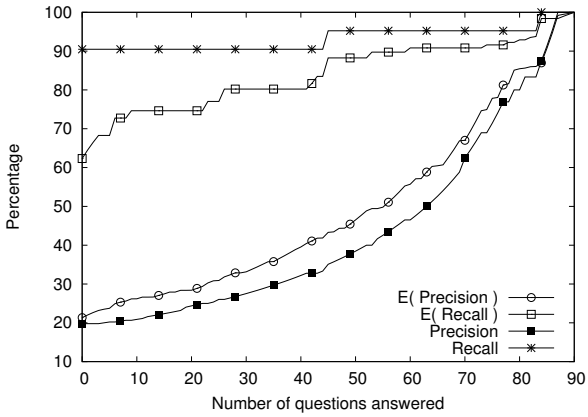

Fig. E.15. Basic DB - Cluster; E (*=PER &1 Verenigde Staten)

E.2 Statistical Strategy

E.2.1 Naive Strategy

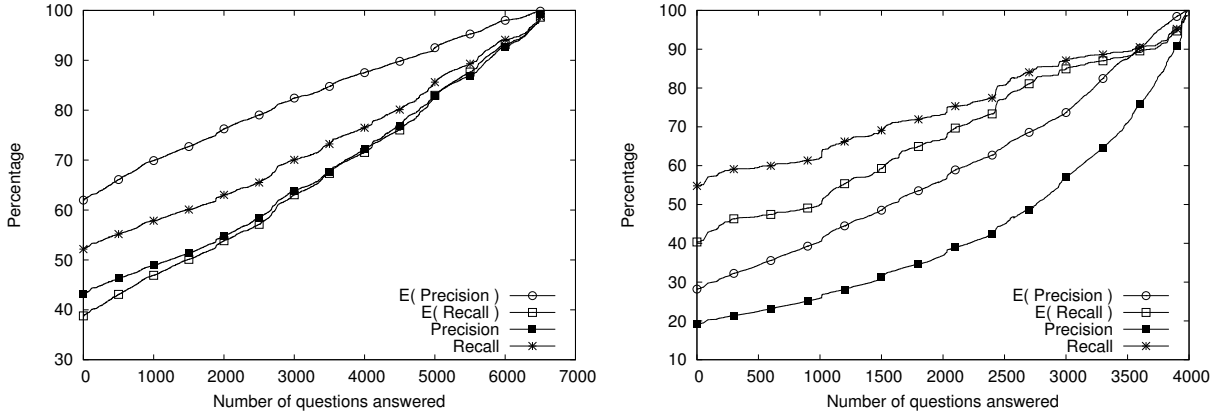

Fig. E.16. Statistical - Naive; Left: A (*); Right: B (*=PER)

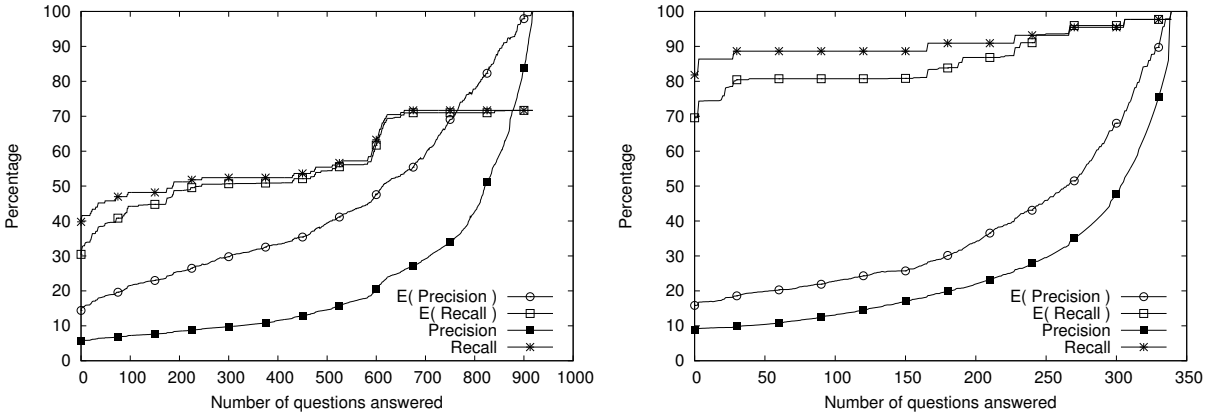

Fig. E.17. Statistical - Naive; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

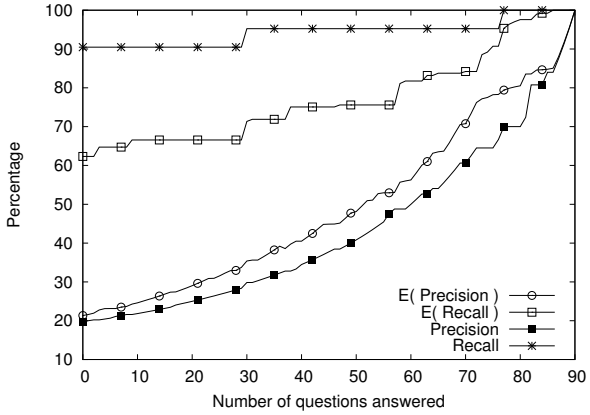

Fig. E.18. Statistical - Naive; E (*=PER &1 Verenigde Staten)

E.2.2 Random Strategy

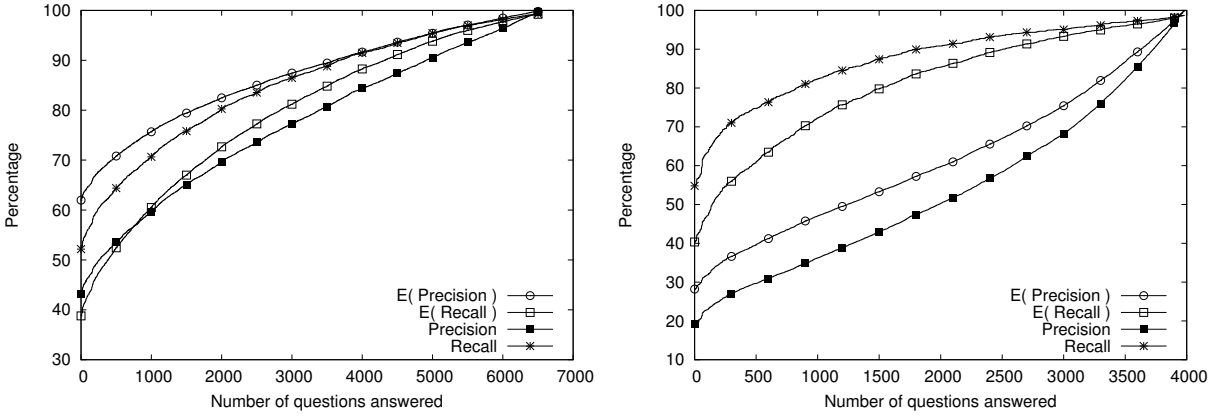

Fig. E.19. Statistical - Random; Left: A (*); Right: B (*=PER)

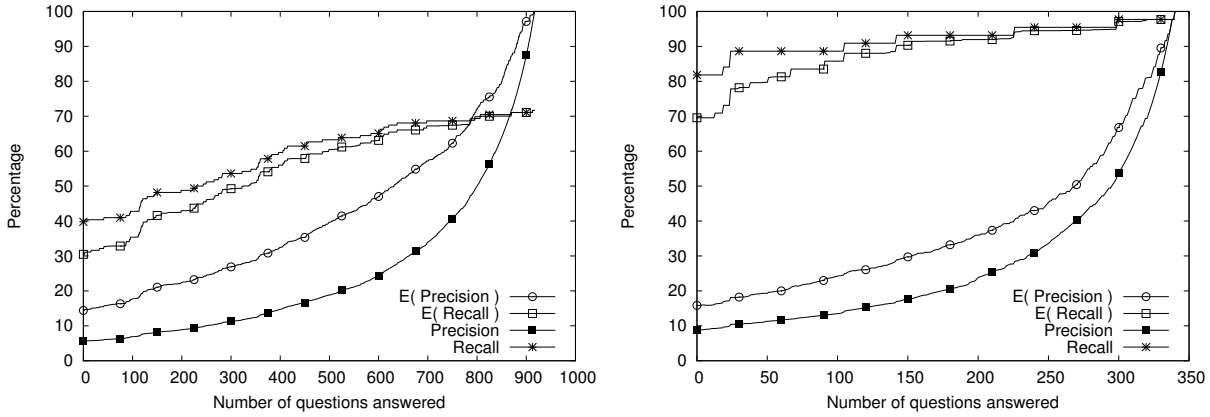

Fig. E.20. Statistical - Random; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

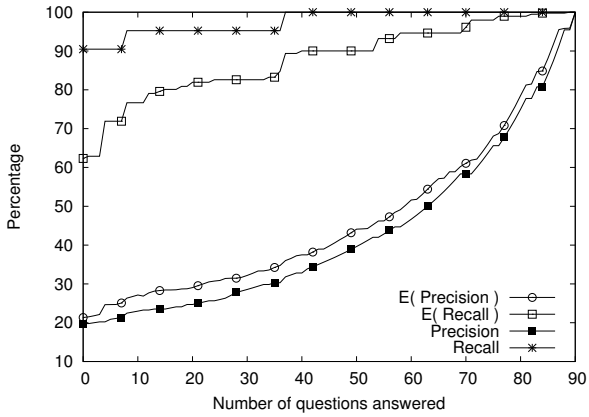

Fig. E.21. Statistical - Random; E (*=PER &1 Verenigde Staten)

E.2.3 Structural Strategy

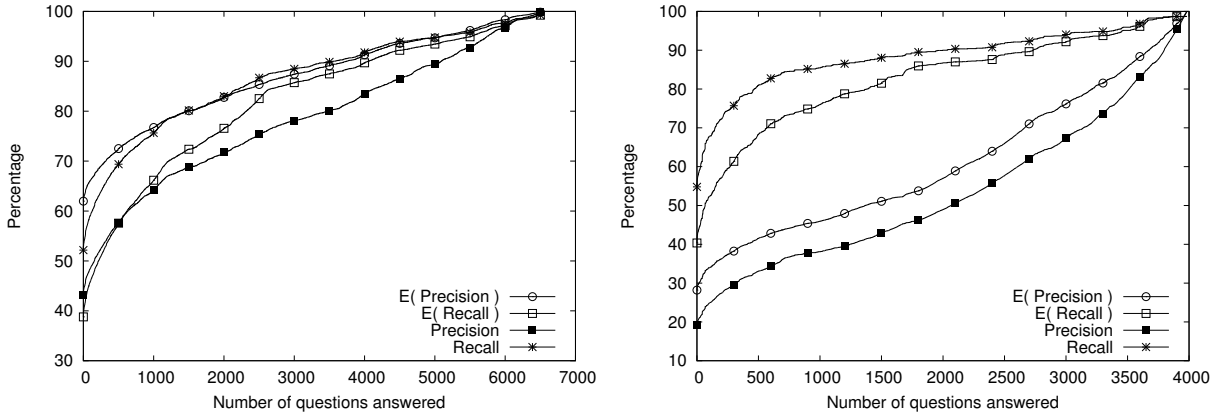

Fig. E.22. Statistical - Structural; Left: A (*); Right: B (*=PER)

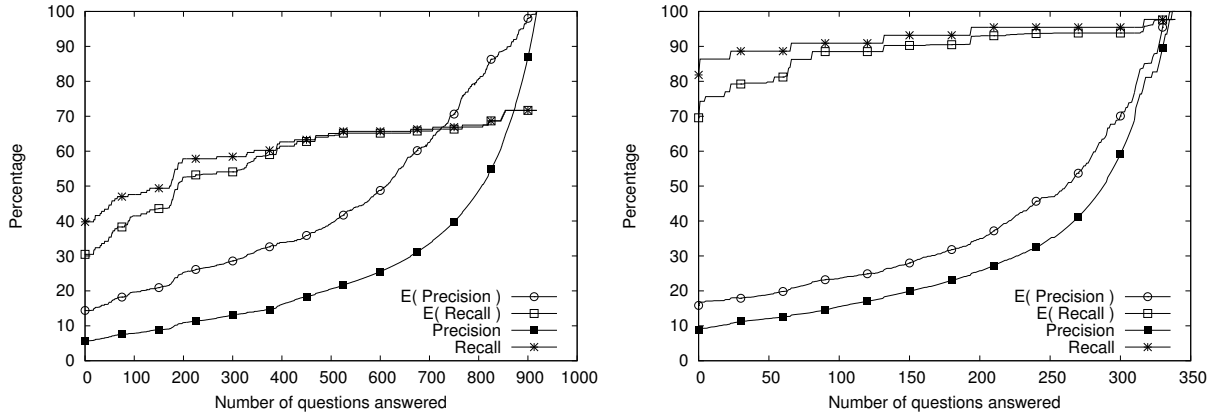

Fig. E.23. Statistical - Structural; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

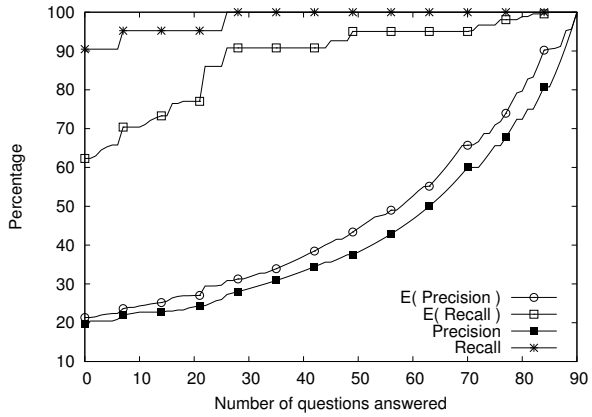

Fig. E.24. Statistical - Structural; E (*=PER &1 Verenigde Staten)

E.2.4 Genius Strategy

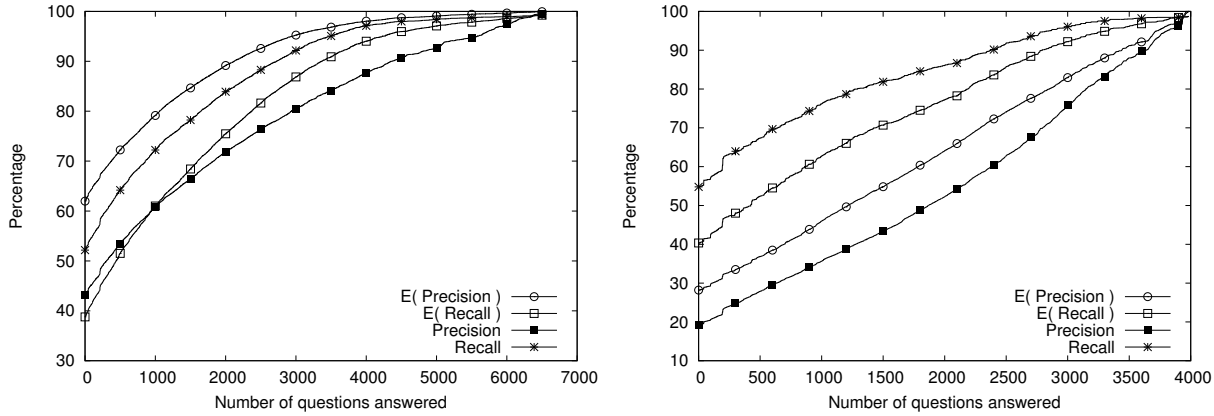

Fig. E.25. Statistical - Genius; Left: A (*); Right: B (*=PER)

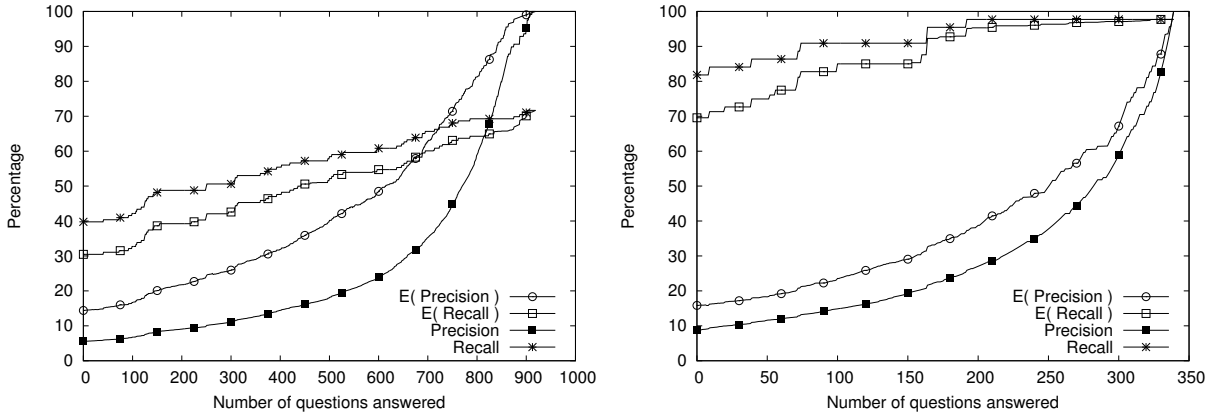

Fig. E.26. Statistical - Genius; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

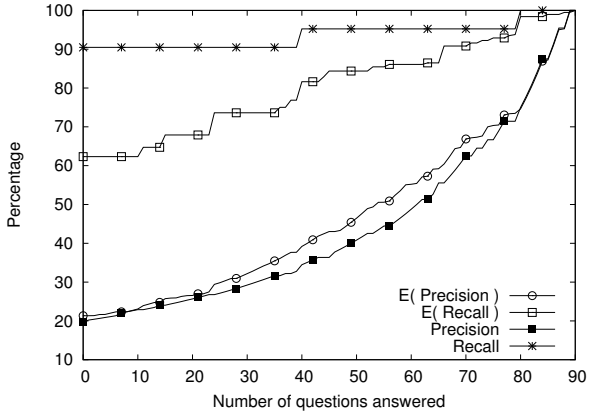

Fig. E.27. Statistical - Genius; E (*=PER &1 Verenigde Staten)

E.2.5 Cluster Strategy

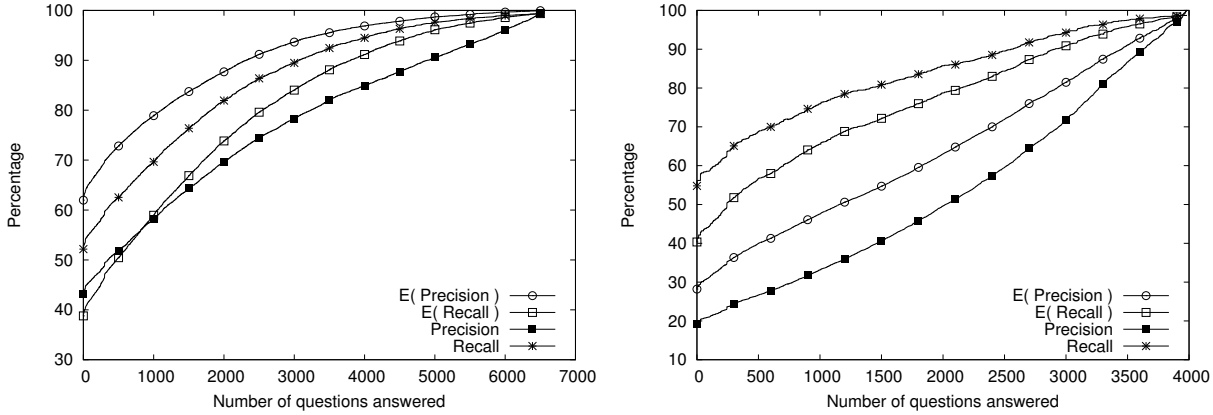

Fig. E.28. Statistical - Cluster; Left: A (*); Right: B (*=PER)

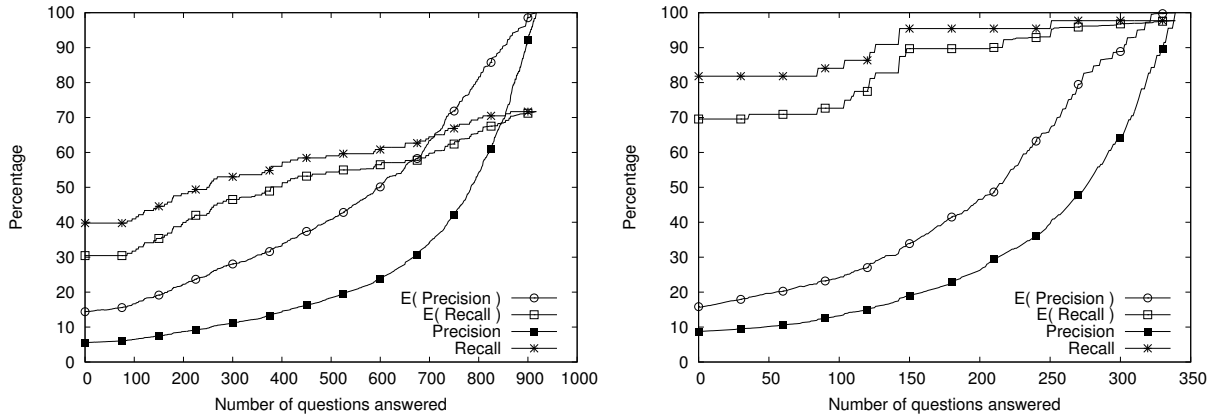

Fig. E.29. Statistical - Cluster; Left: C (*=EVE); Right: D (*=LOC &2 *Dylan*)

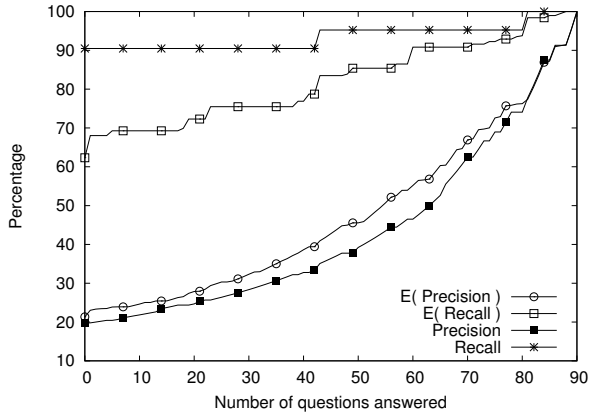

Fig. E.30. Statistical - Cluster; E (*=PER &1 Verenigde Staten)