



**Technische Universität Hamburg-Harburg  
Institut für Softwaresysteme**

# A Strategy to support Information Extraction from Natural Language at production time

Diplomarbeit

Andrey Galochkin

December 2008

1. Gutachter: Prof. Dr. rer.-nat. habil. Ralf Möller
2. Gutachter: Prof. Dr.-Ing. Rolf-Rainer Grigat

# Abstract

On the vision of intelligent authoring tools and question answering systems, it is of relevance to find a strategy that accelerates the process of IE, such that it can occur on the fly.

This thesis presents a novel rule-based system which extracts surface-level information from domain-specific texts using lightweight natural language processing techniques. A practical implementation of this system is provided as a component of the General Architecture for Text Engineering (GATE). The system achieves the F-measure of over 84% on the named entity recognition task with the processing speed of around 65 kilobytes of text per second.

Keywords: information extraction, natural language processing, GATE framework, named entity recognition, athletics domain.

# Declaration

Hereby, I declare that this master thesis has been prepared by myself. All literally or content related quotations from other sources have been pointed out and no other sources than declared have been used.

Hamburg, 19 December 2008

Andrey Galochkin

# Acknowledgements

I would like to thank Prof. Dr. Ralf Möller for providing me with a really interesting and challenging topic of research and giving valuable advice throughout the project.

Further I want to thank Irma Sofia Espinosa Peraldi for her patience, dedication and many long and valuable discussions that provided me with inspiration and guidance throughout this work.

Finally, I would like to thank Kamil Sokolski for helping me getting started with the GATE framework.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
<b>2</b>	<b>Information Extraction .....</b>	<b>2</b>
2.1	Motivation .....	2
2.2	Definition .....	2
2.3	Characteristics of IE .....	2
2.4	Information Extraction tasks .....	3
2.5	Specificity vs Complexity .....	4
2.6	Named Entity Recognition .....	4
2.7	Co-reference Resolution.....	5
2.8	Template Relation Production.....	5
2.9	Ontology-based IE.....	6
<b>3</b>	<b>GATE Description .....</b>	<b>7</b>
3.1	Resources in GATE.....	7
3.2	Applications and Datastores.....	8
3.3	Annotations .....	8
3.4	Built-in Processing Resources.....	9
3.5	JAPE overview .....	9
<b>4</b>	<b>The Developed System.....</b>	<b>12</b>
4.1	Functionality Description .....	12
4.1.1	Input Text.....	12
4.1.2	Named Entities to recognize .....	12
4.1.3	Relations to extract .....	13
4.1.4	Operation Modes.....	14
4.2	Analysis Mode.....	14
4.2.1	Overview .....	14
4.2.2	Preprocessing .....	15
4.2.3	Overview of NE recognition grammars .....	20
4.2.4	NE_before_coref.....	21
4.2.5	Co-reference resolution.....	24
4.2.6	NE_after_coref.....	25

4.2.7	Relation Extraction .....	31
4.3	Training Mode.....	36
4.3.1	The need for adaptability .....	36
4.3.2	Collect ontology concept names .....	37
4.3.3	Collect Gazetteer lists .....	37
4.3.4	Create Test Documents .....	37
<b>5</b>	<b>Performance evaluation .....</b>	<b>42</b>
5.1	Test setup.....	42
5.2	Evaluation metrics .....	43
5.2.1	Definitions.....	43
5.2.2	Corpus Benchmark Tool .....	44
5.3	Results .....	44
5.3.1	Training set .....	44
5.3.2	Evaluation set.....	45
5.3.3	Processing speed .....	46
5.4	Interpretation of results .....	47
5.4.1	Named entity recognition.....	47
5.4.2	Relation Extraction .....	49
<b>6</b>	<b>Discussion .....</b>	<b>50</b>
6.1	Comparison to other projects .....	50
6.1.1	NE recognition .....	50
6.1.2	Relation extraction .....	50
6.1.3	Classification of the System.....	50
6.2	Robustness to changes in the Ontology.....	51
6.2.1	Portability to a new Domain .....	52
<b>7</b>	<b>Conclusions.....</b>	<b>53</b>
7.1	Advantages .....	53
7.2	Disadvantages.....	53
<b>8</b>	<b>Future Work .....</b>	<b>54</b>
<b>A</b>	<b>Appendix.....</b>	<b>55</b>
A.1	List of documents in the corpora.....	55
A.1.1	Training set .....	55
A.1.2	Evaluation set.....	55
A.2	Log of the relation extraction example .....	56
A.3	Evaluation Set results .....	59
A.4	JAPE Rules for named entity “Performance” .....	73
	<b>Bibliography .....</b>	<b>79</b>

# Table of Figures

Figure 1: Performance trade-off relative to specificity and complexity. (Reprinted from [5]) .....	4
Figure 2: Simplified annotation Graph. (Adapted from [11]) .....	9
Figure 3: Relations between the named entities.....	13
Figure 4: Primitive concepts (red) and aggregates (blue) that can be formed using abductive reasoning (Reprinted from [12]). .....	14
Figure 5: Tokeniser processing results.....	15
Figure 6: A list of rankings, correctly parsed as one sentence. ....	17
Figure 7: Tokens after POS tagging and morphological analysis.....	18
Figure 8: Results of Gazetteer lookup. ....	18
Figure 9: Relations between NE extracted from the running example .....	34
Figure 10: Named Entities and relations between them after adding owl:sameAs relation between PersonName1 and PersonName2 .....	36
Figure 11: Processing resources used in the training mode.....	37
Figure 12: An automatically generated test document for the named entity "Stadium" .....	41
Figure 13: Headers and hyperlinks were not annotated by humans, which results in "false positives" (red) .....	42
Figure 14: Average F1-measure for each NE type in the training set and evaluation set .....	46
Figure 15: "Olympic" is a very common descriptor of the NE "SportsEventName". However, human annotations are highly inconsistent here, which leads to many false positives. ....	48

# Table of Tables

Table 1: Named Entity recognition grammars. ....	20
Table 2: Performance achieved on the training set .....	44
Table 3: Performance achieved on the evaluation set.....	45
Table 4: Processing times when ontology is cleaned before processing each document.....	47
Table 5: Processing times without cleaning the ontology .....	47



# 1 Introduction

The aim of this thesis is to develop a software application which extracts surface-level information from domain-specific texts. More precisely, the application should recognize Named Entities (NE), carry out co-reference resolution and extract relations between the NE on texts from the Athletics domain.

The System will be designed and implemented as an application for the GATE framework [7]. Most algorithms will be rule-based, implemented in the JAPE language [8]. The System will take as input a corpus of texts and an ontology in order to extract surface-level information, populate the ontology, and output the resulting analysis ABoxes [29] to a file.

The performance of the system will be evaluated using the Corpus Benchmark tool [7]. Finally, the system will be compared to other projects.

The rest of this thesis is organized as follows:

- Chapter 2 gives an overview of Information Extraction
- Chapter 3 introduces the GATE framework
- Chapter 4 describes the developed system
- Chapter 5 is about performance evaluation and interpretation of results
- Chapter 6 contains the comparison to other projects
- Chapter 7 summarizes the lessons learnt
- Chapter 8 presents the ideas for future work
- Appendix contains the detailed evaluation results and some source code

## 2 Information Extraction

### 2.1 Motivation

The amount of data available nowadays is enormous. From [26] it can be estimated that there were about 14 exabytes of new information produced in 2008. Almost 1% (1400 terabytes) of this information is textual.

To get the relevant information one can use Information Retrieval (IR) techniques or simple keyword search provided by a web search engine such as Google. However, after locating the *documents* of interest much time is usually spent on further reading and analyzing of the texts in order to extract the *facts* of interest.

For example, a user of an IR system wanting information on “pole vault events in Hamburg in the last 5 years” would enter a list of relevant words and receive in return a set of *documents* (e.g. news articles from [www.iaaf.org](http://www.iaaf.org)) which contain the likely matches. The user would then read the documents and extract the requisite information themselves. They might then enter the information in a spreadsheet and produce a chart for a report or presentation.

In contrast, an IE system would populate the spreadsheet directly with the relevant data.

### 2.2 Definition

According to [5], Information Extraction (IE), is “the process of deriving disambiguated quantifiable data from natural language texts in service of some pre-specified precise information need”.

### 2.3 Characteristics of IE

The main characteristics of the IE process are: [11]

*It works with unstructured sources* such news articles, technical manuals, legal documents, speech transcripts or radio broadcasts. All these texts have no predefined structure - they are mostly in natural language form.

*It locates the relevant information* and ignores the non-relevant. As mentioned earlier finding documents that are relevant to some predefined problem is not sufficient because the texts should additionally be read and analyzed. IE tries to extract facts from the text that are relevant to the problem.

*It extracts the information in a predefined format* and the result is a structured representation of the facts of interest available in the texts. The eXtended Markup Language (XML) is the most common format.

*It is usually domain dependent* - an IE system is usually designed and trained to deal with texts that are specific to some domain. Some of the patterns and rules the system will employ are not always applicable to a different problem. For example a system that is designed to extract facts about bankrupting companies from news articles will perform with much lower precision on the news from the athletics domain. Building IE systems that are relatively domain independent or could be ported to different domain more easily is still an open issue.

## 2.4 Information Extraction tasks

The most common tasks performed by IE systems, following the classification from the seventh (and final) Message Understanding Conference (MUC), are: [1]

- Named Entity recognition (NE) - finds the entities in the text
- Coreference Resolution (CO) - finds identities between entities
- Template Element construction (TE) - finds the attributes of the entities
- Template Relation construction (TR) - finds the relations between entities
- Scenario Template construction (ST) - finds the events in which entities participate

Consider these sentences:

*An amazing world record was set by Wilfred Bungei on July 26th. The Kenyan ran 800m in under 1:43:72.*

NE discovers that the entities present are “world record”, “Wilfred Bungei”, “July 26th”, “Kenyan”, “800m” and “1:43:72”.

CO discovers that “the Kenyan” refers to “Wilfred Bungei”.

TE discovers that the “world record” is “amazing”.

TR discovers that “1:43:72” is performance in “800m” and this performance belongs to “Wilfred Bungei”.

ST discovers that there was an athletics event (competition in running 800m) in which the various entities were involved.

## 2.5 Specificity vs Complexity

The quality of the results produced from each task strongly depends on the quality of the results produced from the tasks performed before it (the 'snowball effect'). While the precision of the Named Entity task is usually very high, the quality of the results from the Scenario Template task is usually low. In most cases the tasks of ST and TE are accomplished through abductive reasoning as described in [23, 29].

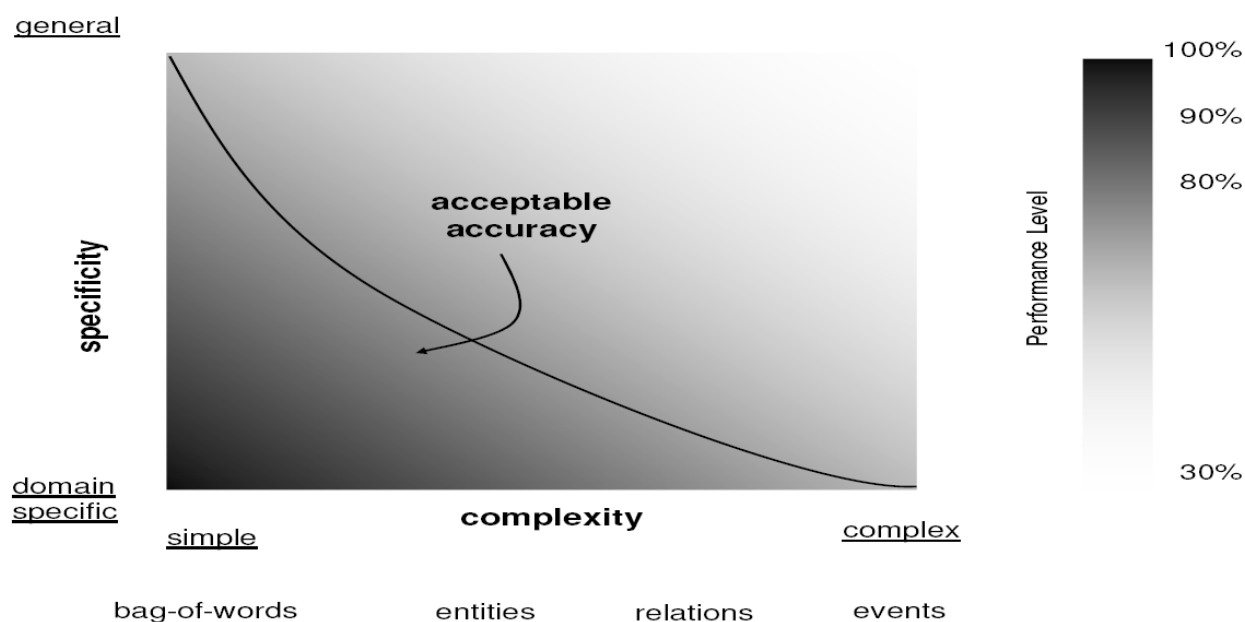


Figure 1: Performance trade-off relative to specificity and complexity. (Reprinted from [5])

The objective of this thesis is to perform NE, CO, and TR tasks on texts in the athletics domain.

A more detailed description of the tasks follows. It is mostly based on [5, 11, 31].

## 2.6 Named Entity Recognition

During the Named Entity recognition phase an IE system tries to identify all mentions of proper names and quantities in the text such as:

- names of persons, locations and organizations
- dates and times
- mentions of monetary amounts and percentages

The relative percentage of the three types of named entities depends on the domain of the texts analyzed. Usually proper names account for 70-80% of the named entities, dates and times account for 10-20% and the number of quantities mentioned in the text is less than 10% [11].

Note that the definition of an entity is domain dependent. It may be a person/organization /date/percentage in the case of analyzing financial news articles, but it may for example be a product name (if texts being analyzed are product catalogs) or a protein/molecule name (in the case of scientific texts).

The quality of the results produced from the NE task is usually very high - the best F-Measure<sup>1</sup> achieved from an IE system in the MUC-7 competition was 93% while humans achieved 98%.

As noted in [14], this task is weakly domain dependent - changing the type of the texts being analyzed may or may not induce degradation of the performance levels.

For example, changing the subject-matter of the texts being processed from financial news to other types of news would involve some changes to the system, and changing from news to scientific papers would involve quite large changes.

## 2.7 Co-reference Resolution

This process identifies whether two or more text portions refer to the same named entity. The following subtasks are of relevance: [31]

*Name matching:* This involves the retrieval of names referring to the same entity (e.g. "Tatiana Lebedeva", "T. Lebedeva", "Lebedeva").

*Pronoun-antecedent co-reference:* Pronouns like 'he', 'she', 'they', 'them', etc. must be associated with their antecedents, resolving them to a domain relevant named entity if possible.

*Definite description co-reference:* This type of co-reference would hold between 'Tatiana Lebedeva' and 'the athlete', or 'Tatiana Lebedeva' and 'The Olympic Gold Medal Champion'.

## 2.8 Template Relation Production

The template relation task requires the identification of a small number of all possible relations between the named entities.

This might be, for example, "has\_start\_date" relation between a date and sports event, "has\_nationality" relation between a person and a country, or "located\_in" relation between a stadium and a city.

These relations can be used for template element construction.

For example, if we define a "high-level concept" Athlete as an aggregate of PersonName, Age, and Nationality, then the extracted relations "personNameToAge" and "personNameToCountry" could correspond to the properties "hasAge" and "hasNationality" of Athlete. (This approach was taken in the BOEMIE project for training purposes with the tool Ellogon.) [10]

---

<sup>1</sup> The F-measure metric will be explained in detail in the Chapter „Performance Evaluation“

Extraction of relations among entities is a central feature of almost any information extraction task, although the possibilities in real-world extraction tasks are endless. In general good TR scores reach around 75%. TR is a weakly domain dependent task.

## 2.9 Ontology-based IE

It must be noted that independently of the techniques used, IE is a knowledge-driven process. It is not just a retrieval or filtering method based on keywords and pattern matching techniques, since the extracted information is interpreted according to a model of the domain-specific knowledge [1].

We could then characterize IE as an ontology-driven process, since an ontology is a formal description of conceptual knowledge for a specific domain.[10] One of the main challenges is the automatic population of ontologies, by identifying instances in the text belonging to concepts in the ontology, and adding these instances to the ontology in the correct location.

In order to automatically perform the IE tasks described in this chapter, natural language engineering (NLE) tools can be used. The next chapter will give a brief introduction to GATE, a NLE framework used in this thesis.

## 3 GATE Description

In this chapter we describe GATE, a General Architecture for Text Engineering, which was used for research in this thesis. Most parts of this description are taken from [11, 8] and [3].

The system developed in this thesis can be used as a GATE application (or standalone application using GATE API) for IE from texts in the Athletics domain. Most of the developed code can be reused or easily adapted for other domains.

GATE is the General Architecture for Text Engineering - a language-engineering environment developed by the University of Sheffield. Since its first release in 1996, this tool was used in a large number of IE applications and gained wide acceptance by scientific community [3].

GATE provides a software infrastructure for NLP researchers, which is made up of three main elements:

- an *architecture* for describing the components composing a language processing system,
- a *framework* (Java class library and API) that could be used as a basis for building such systems,
- a graphical *development environment* (built on top the framework) comprising of a set of tools and components for language engineers.

GATE is designed to separate cleanly low-level tasks such as data storage, data visualization, location and loading of components and execution of processes from the data structures and algorithms that actually process human language [6].

### 3.1 Resources in GATE

GATE distinguishes the following three types of resources:

- *Language Resources* (LR) representing documents, lexicons, corpora, ontologies, etc.
- *Processing Resources* (PR) representing components that operate on language resources such as tokenisers, POS taggers, parsers, etc.
- *Visual Resources* (VR) representing GUI components that visualize and edit the language and processing resources

These three types of resources are used to model the components comprising a language processing system based on GATE. The set of processing resources that are integrated with the system is called CREOLE, the Collection of Reusable Objects for Language Engineering. Language engineers can develop their own resources and integrate them into the GATE system in a plug-and-play manner (details are available in the GATE User Manual [8])

## 3.2 Applications and Datastores

Two other abstractions provided by GATE for the language researchers are *applications* and *datastores*. Applications are sets of processing resources grouped together and executed in sequence over some language resource. Datastores provide persistent representation for language resources.

## 3.3 Annotations

When processing resources (such as parsers, tokenisers, taggers, etc.) that are part of some NLP system operate on the original texts, they produce information about this text. Such information, for example, is the type of a specific token (word, number, punctuation, etc.) that is generated from the tokeniser, or the part of speech for the word (proper noun, pronoun, verb, etc.) that is generated from the POS tagger. The information about the text is usually represented as sets of *annotations* [16].

The approach to storing annotations, accepted by GATE, is the *reference annotation* model. The reference model stores annotations in *annotation graphs* and this information is not embedded in the original text but instead the generated annotations refer to fragments of the text by reference [6].

A GATE annotation consists of:

- ID, which is unique in the document the annotation refers to
- type, which denotes the type of the annotation (different processing resources usually generate annotations of different types)
- start and end node, which denote the span of the annotation, i.e. the fragment of the original text the annotation refers to
- a set of features (attribute/value pairs) that provide additional information about the annotation

For example, for the following sentence:

*Mr. Obama is the President of the United States.*

...an annotation graph like this one will be created:



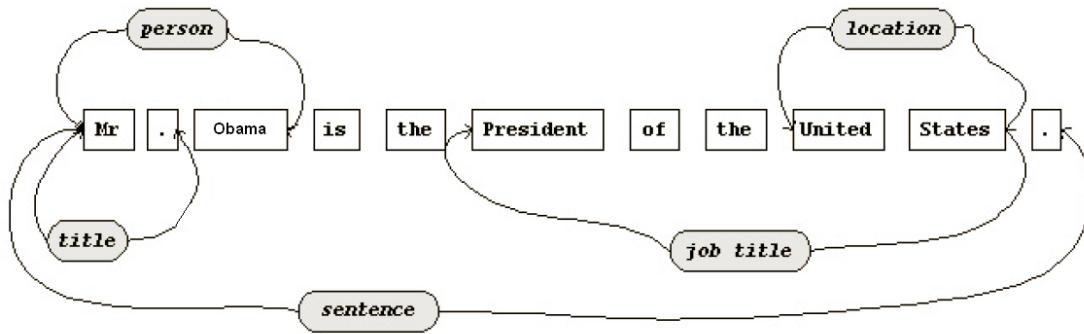


Figure 2: Simplified annotation Graph. (Adapted from [11])

Note that this is not the full annotation graph, most of the annotations are actually skipped for simplicity. The complete annotation graph for the sentence contains more than 30 annotations, together with their features.

### 3.4 Built-in Processing Resources

GATE contains a set of built-in components called ANNIE (A Nearly-New IE system). Some of the most important components that comprise ANNIE are:

- tokeniser
- gazetteer
- sentence splitter
- POS tagger
- named entity transducer
- orthographic name-matcher

The description of these resources and all modifications made to their default version for the purpose of this thesis will be provided in the next chapter.

### 3.5 JAPE overview

JAPE is the Java Annotation Patterns Engine in GATE. It provides finite state transduction based on regular expressions over GATE annotations [6].

Each JAPE transducer is provided with a JAPE grammar and if the rules defined in the grammar are satisfied by the annotations generated so far in the system it will perform the action which is specified in the grammar for the rule.

A JAPE grammar consists of a set of *phases*, which are executed sequentially. A phase consists of:

- unique (for the grammar) name
- input specifier, which defines the annotation types that are considered as valid input for the phase

- options specifier, which modifies the behaviour of the JAPE engine when executing this phase
- one or more *macros* (optional)
- one or more *rules*

A rule consist of:

- unique name
- optional priority
- left-hand-side (LHS) specifying a regular expression to be matched against annotations
- right-hand-side (RHS) specifying the action to be performed if the rule is satisfied

Since the LHS of a rule is a regular expression, it can contain regular expression operators such as "\*", "?", "|" or "+". The RHS may contain any valid block of Java statements, which makes it quite powerful and flexible.

A detailed overview of the JAPE engine and the BNF for JAPE grammars is available in [8].

We will present only a simplified example.

A sample JAPE grammar is:

```
Phase: test
Input: Token
Options: control=appelt

Rule: LocCity
Priority: 30
(
  ({Token.category == NNP})[1,2]
  {Token.string == "City"}
):loc
-->
:loc.City = {rule="LocCity"}
```

The sample grammar begins with definition of a single phase. The name of the phase is specified together with the annotations that the phase will accept as a valid input. In this case only Token annotations will be matched against the rules and all other annotations already generated in the system will be ignored.

Additionally, the options of the phase instruct the JAPE engine to run in "appelt" mode, which means that if several rules are satisfied by the input at some moment (i.e. their LHS are matched by the input), then only the longest matching rule will be applied (the one that matches the longest sequence of annotations from the input). If there are several rules that match a region of the input with the same length, then rule priority is considered and the rule with the highest priority will be applied. Another style of processing is "brill" and when the engine runs in this mode then all the rules that are satisfied are applied (and respectively the priorities assigned to the individual rules are ignored). The final style is "first" which instructs the JAPE engine to apply the first rule that matches the input region.

The "LocCity" rule has a priority set to 30 (priorities are useful only for "appelt" style of the phase). The LHS of the rule (the block in brackets preceding the "-->" symbol) contains a regular expression that will be matched against the input (Token)

annotations. The rule says that the sequence of one or two proper noun tokens and the tokens "City" should be matched. (This is because the "string" feature of Token annotations contain the actual word or symbol of the token and the "category" feature contains the part-of-speech of the token assigned by the POS tagger). The matched sequence of input annotations can later be referred to as "loc".

Finally, the statements in the RHS (the block in brackets following the "-->" symbol) will be executed whenever the LHS is satisfied by the input. The RHS will create a new annotation of type City for the span of the matched annotations. The City annotation will have a single feature "rule" that will be set to "LocCity".

If we apply the rule to the input sequence of Token annotations "New", "York" and "City", a new City annotation will be created for the span "New York City".

# 4 The Developed System

## 4.1 Functionality Description

The System takes as input a corpus of texts and an ontology in order to extract surface-level information, populate the ontology, and save the resulting analysis ABox [29] to a file for each text in the corpus.

### 4.1.1 Input Text

Most texts used in this thesis for training and evaluation purposes were collected from the following websites, which contain an archive of daily news on athletics: [12]

<http://www.iaaf.org>, <http://www.european-athletics.org>, <http://www.usatf.org/>,  
<http://www.ukathletics.net>.

The list of documents in the training and evaluation corpus are located in the Appendix.

### 4.1.2 Named Entities to recognize

In the scope of this thesis the following NE from the athletics domain need to be recognized [10] :

- *PersonName*: the name of an athlete.
- *Gender*: the gender of an athlete (male or female).
- *Age*: the age of an athlete.
- *Ranking*: the ranking of an athlete during a specific sports trial in a specific athletics competition.
- *Performance*: the performance of an athlete during a specific sports trial in a specific athletics competition.
- *Country*: the name of the country where a sports event might have taken place, or the nationality of an athlete.
- *SportsEventName*: the name of a sports event, such as the “Olympic Games”.
- *SportsName*: name of a specific sports competition , such as “Pole Vault Competition”.
- *SportsRoundName*: the name of a specific round of a sports competition , such as “semifinal”, “final”, etc.
- *City*: the name of the city where a specific event has taken place.
- *Date*: the date of a specific athletic event
- *Stadium*: the name of the stadium, where a specific sports event takes place.

Each of these entities corresponds to a “primitive concept” of the ontology, i.e. atomic concept that occurs only on the right-hand side of terminological axioms [9].

By design this mapping is unique.

### 4.1.3 Relations to extract

Figure 3 shows some of the most important relations between the NE that need to be extracted. An arrow connecting two NE represents a useful relation between them.

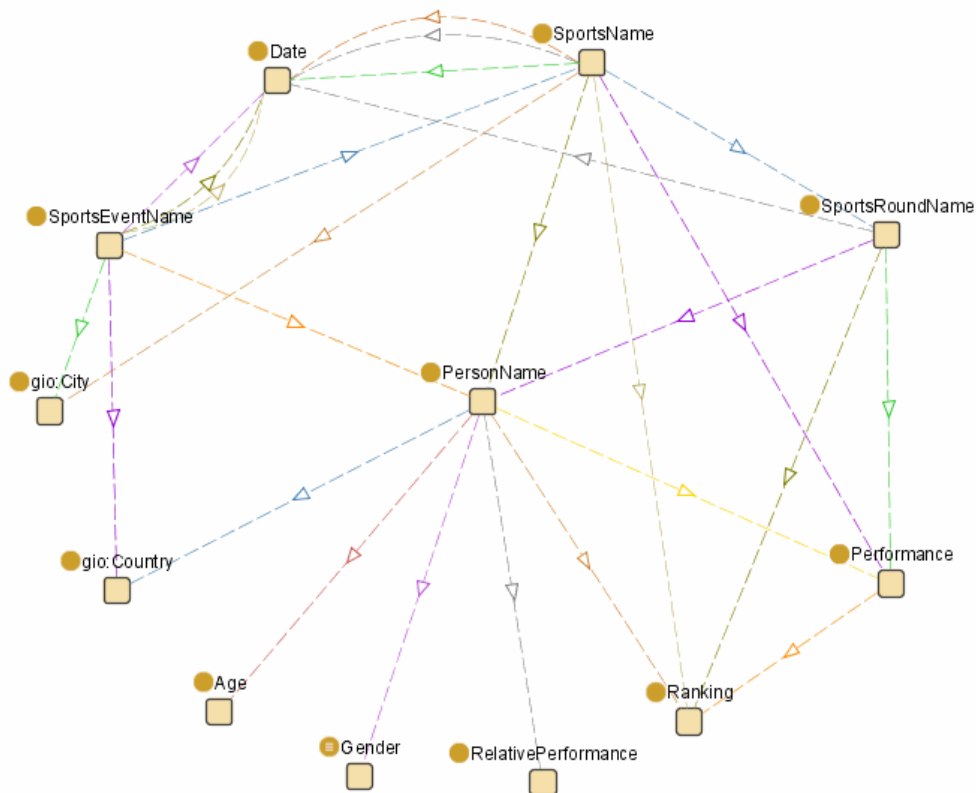


Figure 3: Relations between the named entities

The roles defined in the ontology are those relations that give a significant contribution to form an aggregate (template element). For example, the named entities PersonName, Age, Gender and the relations PersonNameToAge, PersonNameToGender can be used to form an aggregate “Person”, as described in the chapter on Information Extraction and in [29].

On the other hand, relations such as AgeToCity are not relevant for the Athletics domain to form an aggregate. Therefore, they are not defined in the ontology.

Figure 4 shows some of the most important aggregates that can be formed by the detected named entities and their relations. The NE are printed in red, aggregates in blue.

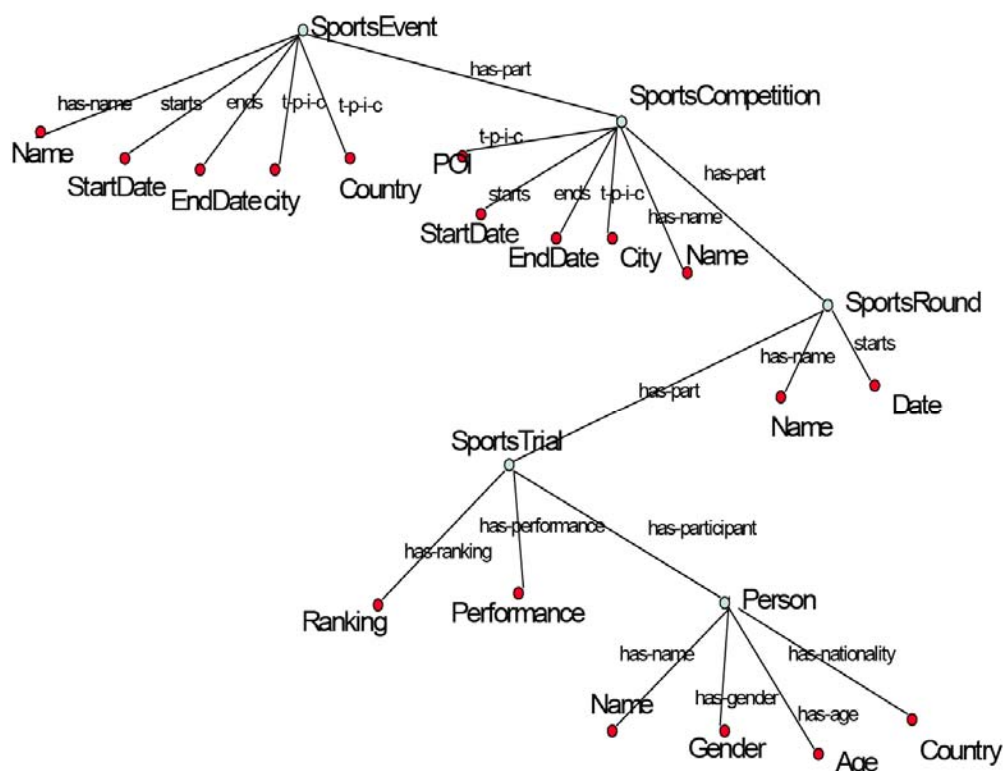


Figure 4: Primitive concepts (red) and aggregates (blue) that can be formed using abductive reasoning (Reprinted from [12]).

By design, the mapping between primitive concepts and the corresponding named entities is unique.

#### 4.1.4 Operation Modes

The *Analysis Mode* of operation applies each time a new multimedia document becomes available. The aim of the analysis mode is to extract NE, perform coreference resolution, populate the ontology with NE, extract relations between NE, and to output the resulting analysis ABox to a file for each document in the corpus.

The *Training Mode* is used to train the System when manually annotated data becomes available. This mode also generates training documents, which facilitate system development and testing.

## 4.2 Analysis Mode

### 4.2.1 Overview

First, the input text is preprocessed using Tokeniser, Sentence Splitter, POS tagger, Morphological analyzer and Gazetteer processing resources.

After that, NE recognition is performed for NE, for which co-reference resolution need to be carried out. Next comes the co-reference resolution step, followed by a second NE recognition Multiphase.

Finally, Ontology is populated with the extracted NE, relations between them are extracted and the resulting analysis ABox is written to a file.

The NE are recognized in two passes because some NE grammars depend on the previously recognized named entities. For example, in the string “Bungei (24)”, the token “24” can only be recognized as Age if “Bungei” was previously recognized as PersonName.

## 4.2.2 Preprocessing

Preprocessing involves modules for the extraction of lexical and syntactic information from text. More specifically, preprocessing involves modules for: tokenization, sentence splitting, POS-tagging, lemmatizing, gazetteer lookup. The descriptions of these components follow.

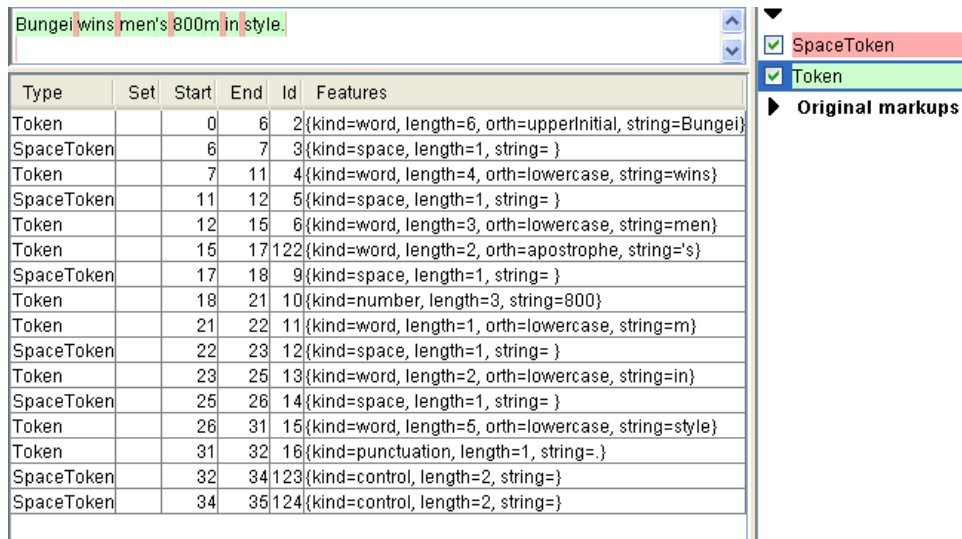
To demonstrate the functionality of the processing resources described below, let's assume that the following text is given as input to the system:

Bungei wins men's 800m in style.

This example will be used for illustration purposes throughout this chapter.

### Tokeniser

This component splits the text into simple Tokens such as punctuation, numbers, and words of different types (for example: words in uppercase and lowercase). It also produces SpaceTokens, which separate Tokens from one another:



Type	Set	Start	End	Id	Features
Token		0	6	2	{kind=word, length=6, orth=upperInitial, string=Bungei}
SpaceToken		6	7	3	{kind=space, length=1, string= }
Token		7	11	4	{kind=word, length=4, orth=lowercase, string=wins}
SpaceToken		11	12	5	{kind=space, length=1, string= }
Token		12	15	6	{kind=word, length=3, orth=lowercase, string=men}
Token		15	17	122	{kind=word, length=2, orth=apostrophe, string='s}
SpaceToken		17	18	9	{kind=space, length=1, string= }
Token		18	21	10	{kind=number, length=3, string=800}
Token		21	22	11	{kind=word, length=1, orth=lowercase, string=m}
SpaceToken		22	23	12	{kind=space, length=1, string= }
Token		23	25	13	{kind=word, length=2, orth=lowercase, string=in}
SpaceToken		25	26	14	{kind=space, length=1, string= }
Token		26	31	15	{kind=word, length=5, orth=lowercase, string=style}
Token		31	32	16	{kind=punctuation, length=1, string=.
SpaceToken		32	34	123	{kind=control, length=2, string=}
SpaceToken		34	35	124	{kind=control, length=2, string=}

Figure 5: Tokeniser processing results

GATE provides two Tokenisers: “Unicode” and “English Tokeniser”. The difference between the two is that the English Tokeniser has an embedded JAPE transducer, which allows to carry out post-processing on the detected tokens. The standard

grammar concatenates various tokens to create constructs like "80's", "don't", "1st", "2nd", etc.

However, I did not use this functionality, in order to have more flexibility in the design of NE recognition rules. For example, descriptors of the NE "Ranking" can be Ordinals such as 1st, 2nd, etc. By annotating only the digits, we can achieve some normalization of the NE during recognition. For example: "He was 1st" → Ranking="1". This would not be possible if "1st" was one whole token.

Next design decision was to choose the rule set for the Tokeniser. The alternate rule set treats hyphens as separate tokens, whereas the default rule set does not. For example, the string "30-year-old" would be parsed as {"30", "-", "year-old"} by the default rule set and as {"30", "-", "year", "-", "old"} by the alternate rule set. I chose the alternate rule set because it gives more flexibility in the design of NE recognition grammars.

## Sentence Splitter

Like all GATE components, the Sentence Splitter is a finite state transducer, whose behaviour is fully controlled by the underlying JAPE grammars and Gazetteer lists.

The default behaviour of this component is:

- split sentences at FullStop ( ".", "...", "!?" ) or double NewLine character ("\\n\\n"),
- do not split when FullStop follows tokens such as *etc*, *inc*, *prof*.

I found that this version can produce orphan tokens which do not belong to any sentence if applied to html text.

Moreover, when applied to lists of athletics rankings, sentences can be split at wrong places.

For example:

Heat 1 : 1 . Jennifer Coogan , 58.91 ; 2 . Rebecca Forlong , 59.32 ; 3 . Jessica Doorey

Men's London Marathon:

1. Felix Limo (Kenya) 2 hours 6 minutes 39 seconds
2. Martin Lel (Kenya) 2:06:41
3. Hendrick Ramaala (South Africa) 2:06:55

Therefore, a rule was added which prevent sentence split in these cases:

```
Rule: notFullStop
(
  ( {Token.string == ":"} |
    {Token.string == ";"} | (NEWLINE) )
  ( {SpaceToken} ) *
  {Token.kind == number}
  ( {SpaceToken} ) *
  (FULLSTOP)
):cr
-->
:cr.NoSplit = {rule = "notFullStop"}
```



Now sentences are split correctly, as can be seen in Figure 6:

Type	Set	Start	End	Id	Features
Sentence		0	191	98	{}
NoSplit		21	25	93	{rule=notFullStop}
NoSplit		73	76	94	{rule=notFullStop}
NoSplit		103	106	95	{rule=notFullStop}
NoSplit		146	149	96	{rule=notFullStop}
Split		191	192	97	{kind=external}

Figure 6: A list of rankings, correctly parsed as one sentence.

The advantage of this approach is that all related named entities remain in the same sentence and can be correctly processed by the relation extraction algorithm (see 4.2.7). In the example above, the relations between “London marathon” and each of the athletes names will be recognized.

## POS Tagger

The tagger is a modified version of the Brill tagger, which produces a part-of-speech tag as an annotation on each word or symbol, for example

“Friday” → NNP (proper noun - singular)

“12” → CD (cardinal number)

“jumped” → VBD (verb, past tense)

POS Tags are needed to differentiate between nouns, verbs and other tokens in the NE extraction phase. Moreover, this component is a prerequisite for the Morphological analyzer. The default settings were left unchanged.

## Morphological analyzer

This processing resource is used to obtain lemmas of tokens, which are needed for flexible gazetteer lookup. The Morpher requires that the POS Tagger and Sentence Splitter are run first. The default set of rules for this component was left unchanged.

Type	Set	Start	End	Id	Features
Token		0	6	186	{category=NNP, kind=word, length=6, orth=upperInitial, root=bungei, string=Bungei}
Token		7	11	188	{affix=s, category=VBZ, kind=word, length=4, orth=lowercase, root=win, string=wins}
Token		12	15	190	{affix=s, category=NNS, kind=word, length=3, orth=lowercase, root=man, string=men}
Token		15	16	191	{category=POS, kind=punctuation, length=1, root=',, string=}
Token		16	17	192	{category=VBZ, kind=word, length=1, orth=lowercase, root=s, string=s}
Token		18	21	194	{category=CD, kind=number, length=3, root=800, string=800}
Token		21	22	195	{category=NN, kind=word, length=1, orth=lowercase, root=m, string=m}
Token		23	25	197	{category=IN, kind=word, length=2, orth=lowercase, root=in, string=in}
Token		26	31	199	{category=NN, kind=word, length=5, orth=lowercase, root=style, string=style}
Token		31	32	200	{category=., kind=punctuation, length=1, root=., string=}

Figure 7: Tokens after POS tagging and morphological analysis

I found that for the NE recognition task, lemmas of tokens are more useful than stems, because the stems can cause false positives:

```
Stem("champions") == "champion"
Stem("Championships") == "champion".
```

“Champion“ is a descriptor of *Ranking*, whereas “Championships“ is a descriptor of *SportsEventName*.

### Gazetteer

The Gazetteer is used in order to classify certain types of words, for example: cities, organizations, countries, dates, etc. The gazetteer contains lists of words for which a major type, and an optional minor type are defined. If words on this list are identified in the text, they are annotated with the major-, and minor type:

Type	Set	Start	End	Id	Features
Lookup		0	6	320	{majorType=PersonName}
Lookup		7	11	324	{majorType=Ranking}
Lookup		7	11	325	{majorType=rankingVerb}
Lookup		12	15	326	{majorType=Gender, minorType=Male}
Lookup		18	22	327	{majorType=SportsName}
Lookup		18	22	323	{majorType=SportsName}

Figure 8: Results of Gazetteer lookup.

Most of these lists were collected automatically in the Training Mode, as described in 4.3.3.

### Choice of the processing resource

The GATE framework provides three Gazetteer types to choose from:

- Annie Gazetteer
- OntoGazetteer

- OntoRoot gazetteer.

OntoGazetteer and OntoRoot gazetteer both provide the facility of inserting the detected NEs as Instances in the ontology. For example, if a string “pole vault” is detected in the text, it could be added as an instance of the high-level concept “PoleVault”.

However, this would be wrong in the scope of this project, because the extraction system is supposed to only detect surface-level concepts corresponding to Named Entities. (To be precise, “pole vault” should be recognized as “PoleVaultName”).

Moreover, if several gazetteer lists match the same string in the document, several Instances will be added to the ontology. This is unacceptable, since we want to have a unique mapping between Named Entities and ontology concepts.

In addition, I found that Annie Gazetteer works about 10% faster than Ontogazetteer. Finally, this Processing resource should be case-sensitive, to avoid identifying descriptors like "and", "nice", "bar", "can" as Named Entities “Country” or “City”.

#### ***Lookup both, string and root***

Using Lemmas of tokens for pattern matching gives some flexibility in the design of NE recognition grammars.

However, in many cases the lemmatizing works incorrectly, which results in lemmatized strings not being present in the gazetteer lists:

Cayman Islands → cayman island

Bill Gates → bill gate

Athens → athen

Pavla → pavlum

Therefore, the decision was made to lookup both, the lemma of token and the original string. In some cases this causes duplicate Lookups. However, the impact on the processing speed is negligibly small.

### 4.2.3 Overview of NE recognition grammars

NE_before_coref	Input NE	Output NE	Comment
first	-	-	defines macros
url	-	Url	needed to prevent false positives
number	-	Money, Percent	needed to prevent false positives
removeSpurious	“bad” named entities	-	“deletes” unreliable text
nationality	-	Nationality	needed for personNameToCountry
Country	-	Country	
Stadium	-	Stadium	
City	Country, Stadium	City	
PersonName	Nationality, Country	PersonName	
unknown	all NE	Unknown	any candidates for Orthomatcher

NE_after_coref	Input NE	Output NE	Comment
first	-	-	defines macros again
addClassFeature	all NE	same	add “class”
Gender	PersonName	Gender	class={Male, Female}
Age	PersonName, Gender	Age	
Performance	PersonName	Performance	
Date	PersonName	Date	
SportsNameSubclasses	-	subclasses of SportsName	dynamicList created in the training mode using the Ontology (4.3.2)
SportsName	Gender, City	SportsName	
Ranking	PersonName Performance Age Date	Ranking	
SportsEventName	Ranking SportsName	SportsEventName	
SportsRoundName	all NE	SportsRoundName	
clean			removes spurious NE

Table 1: Named Entity recognition grammars.

#### 4.2.4 NE\_before\_coref

This file contains a list of the grammars used for NE, in the correct processing order. The ordering of the grammars is crucial, because they are processed in series, and later grammars depend on annotations produced by earlier grammars and macros defined in them.

As mentioned earlier, the pipeline does co-reference resolution using the Orthomatcher processing resource. It creates new Annotations when a partial match of an already existing annotation is found. (Currently, co-reference resolution is only carried out for the named entities PersonName and Country, because too many false positives are created otherwise).

These two named entities are used as context for recognition of other NE.

Consider an example:

“Bungei (24yrs) wins 800m.  
Wilfred Bungei from Kenya...”

The string “24yrs” is only recognized as “Age” if it follows a PersonName. However, if the PersonName is not recognized yet, neither can “Age”.

Therefore, the NE recognition grammars were separated in two parts: before and after the Coreference resolution.

##### **first**

This grammar is processed first and defines several macros, which are used in the subsequent grammars, for example SPACE, ONE\_DIGIT, TWO\_DIGIT etc. (In Gate, macros defined or annotations produced by one grammar are available to all subsequent grammars within a single Multiphase)

##### **url**

Slightly modified original ANNIE grammar, which recognizes URL strings in the document based on the prefix “www”, “http” or both.

##### **number**

Recognizes “Percent” and “Money” named entities.

##### **removeSpurious**

Annotations such as Url, Percent, Money can potentially create spurious annotations.

For example, in the URI:

*[www.sportsofworld.com/olympics/athletics/women-20km-walk-2004.html](http://www.sportsofworld.com/olympics/athletics/women-20km-walk-2004.html)*

the grammars would identify the following (spurious) named entities:

*“women”, “20km-walk”, “2004”*

By “deleting” the text corresponding to the URI from the document, we prevent spurious annotations<sup>2</sup>.

The same approach will be applied to the NE Money and Percent. For example, the string “US \$5,000” could be recognized as Country (“US”) or SportsName “5,000 m running”.

## Country

This grammar contains rules to identify Countries via the gazetteer lists “country” and “country\_abbreviation”.

The entries in these lists are unambiguous, (e.g. there are no persons or SportsNames named “Germany”), therefore this NE can be recognized without using context or other NE.

This phase is processed first, and has only one rule.

## Stadium

This phase recognizes Stadium Names, such as “Olympic Stadium” or “Stade de France”.

One rule uses a Gazetteer list created by a Gazetteer list collector from the training document set.

The other rules looks for the keyword “stadium” in the vicinity of capitalized words or Country or City, like “Estadio de Madrid” or “stadium of Chicago”

## City

This phase recognizes City names.

The recognition is mainly based on the Gazetteer list, which was collected in the training mode.

To support NE recognition of previously unseen examples, two rules have been added, which recognize City as “one or two capitalized nouns followed by a comma and a country name”. For example:

Madrid, Spain  
Aix-en-Provence, France  
Bad Kleinen, Germany  
Tsu-Yan, China

To reduce the number of false positives, we create a negative rule, which prevents other NEs from matching the previous rule.

---

<sup>2</sup> The actual implementation is to remove all Tokens, Lookups and NE within the span of the “spurious” annotations. This makes that span of text “invisible” for all other NE grammars. We do not actually delete the “bad” text due to performance reasons. If a text is removed, then all remaining annotations’ offsets need to be updated, which can take long on large texts.

For example:

Claude Richard, France	[PersonName]
21 April, Germany	[Date]
Athletics Stadium, USA	[Stadium]
Olympic Games, China	[SportsEventName]
etc..	

## PersonName

This phase detects Person Names, such as Wilfred Bungei, Osmar Barbosa Dos Santos etc.

This phase heavily relies on the gazetteer list Personname.lst, which is obtained by automatically collecting manual annotations from the training corpus.

In the training phase all personNames were split into individual tokens and added to the gazetteer list separately. For example, “Osmar Barbosa Dos Santos” will be added completely, as well as each comprising token “Osmar”, “Barbosa”, “Dos”, “Santos”.

This is necessary to allow generalization (rules doing it follow).

The only modification is that we remove initials, to prevent spurious matches. (These tokens are deleted from the gazetteer list).

For example, if we have a string “Smith F.S.”, then the Orthomatcher could match the initials “F” and “S” with other letters, such as F (Gender), f (round name “final”) or S like in “Women’s pole vault”.

Athletics news contain many one-letter abbreviations, so it is unsafe to allow person name initials to match.

The most basic algorithm would simply lookup each entry in the Gazetteer list and create a PersonName if a match is found. However, this would cause many false positives. For example, “Jan” could be an abbreviation for the month “January”, “London” could be a City, and Pole could be either Nationality or abbreviation for PoleVault.

The following modifications largely prevent ambiguity.

- only create a PersonName from a single lookup, if it consists of at least two tokens. For example, if the Lookup “Jan Friedrich” is found in the text, we can be sure that “Jan” is part of a person name and not a month. The same holds for “Jack London” or “Pole Anderson”.
- any combination of subsequent PersonNames without commas in between must be one single personName. For example, “Abdula Garcia Dos Santos” is a PersonName consisting of four Lookups in the Gazetteer list.

This simple algorithm ensures that about 97% of all names in the training set are correctly recognized.

However, the performance on previously unseen texts is poor because many names are previously unseen, for example “David Thsuj”.

Therefore, we add the following rule to create a PersonName if the following patterns are matched:

- Lookup followed or preceded by a capitalized Token.

-> If a name is followed by a capitalized Token, then both tokens must be one personName.

For example, in the string “David Thsuj”

This would correctly recognize “David Thsuj” as a person.

However, this modification would also produce many spurious matches such as “Powerful Nelson”, “Kenyan Bungei”.

Therefore, we say that the above rule shouldn’t fire, if the capitalized token is a Nationality, an adjective or an adverb.

Finally, we can be sure that the two tokens are a PersonName, if they are

- followed by a verb (e.g. Lashawn Merritt *jumped*)
- preceded by a nationality (e.g. *Bahrain’s* Youssef Saad Kamel)
- followed by a nationality (e.g. Bayano Kamani *of Panama*)

Last but not least we add rules which recognize a capitalized token as a personName if it is followed by the words “who”, “whose”, “whom”. (From this context we can be sure that the preceding token must be a PersonName).

Yuko, *who* will be running...

Huji, *whose* performance was...

To further reduce the number of spurious annotations we demand that a personName mustn’t be preceded by the token “in” (which is a context for “Location”). For example, if the current string is “in Sesto San Giovanni”, we can be sure that it is not a PersonName.

## unknown

This short grammar finds proper nouns not previously recognized, and gives them an Unknown annotation. This is then used by the namematcher – if an Unknown annotation can be matched with a previously categorized entity, its annotation is changed to that of the matched entity. Any remaining Unknown annotations are useful for debugging purposes, and can also be used as input for additional grammars or processing resources [7].

### 4.2.5 Co-reference resolution

The only type of co-reference resolution implemented so far is orthographic matching of strings, using the GATE component Orthomatcher [8].

This module adds identity relations between named entities. It does not find new named entities such, but it may assign a type to an unclassified proper name, using the type of a matching name. The matching rules are only invoked if the names being compared are both of the same type, i.e. both already tagged as (say) PersonName, or if one of them is classified as ‘unknown’. This prevents a previously classified name from being recategorized.

A lookup table of aliases is used to record non-matching strings which represent the same entity, e.g. “Iceland” and “ICE”, “WR” and “World Record”.



The following NE types can be assumed unique within the same document: PersonName, Country, Stadium, City.

However, after examining them I found out that too many spurious matches are produced if Stadium and City are orthomatched. One reason is that Stadium and City annotations can overlap. E.g. "Wembley" can represent both, a City and a Stadium.

Therefore, coreference resolution is only carried out for "PersonName" and "Country". Ontology instances corresponding to orthomatched NEs will be assigned "owl:sameAs" relation in the relExtraction phase.

## 4.2.6 NE\_after\_coref

### Gender

This Phase recognizes Gender of Persons and assigns the class feature of either Male or Female.

It uses two gazetteer lists, male.lst and female.lst, which contain all possible descriptors for this concept.

For example: female = {woman, women, womens, female, girl, W...}

Negative rule:

Prevent initials in Names like "Smith, W. M." to be recognized as Gender

all other rules are

- based on the gazetteer lists presented above
- independent on other NE
- context-free (i.e. if an entry from the list matches some portion of the text, it is always annotated as Gender)

### Age

This Phase recognizes Age of Persons, which must be an integer number.

First, we define that candidates for Age must be integers with at most 2 digits. (That is, numbers less than 99).

This is necessary to reduce the number of conflicts with other NE, such as Date which can also be an integer, for example, 2004.

Next, we identify relevant context, which is described below.

Almost each phase has rules independent and dependent on other NE.

For best clarity, I show the typical patterns that the system can detect.

- Tokens that should be assigned the annotation label are underlined.
- tokens used as context are in italics.

To make the examples more understandable, sometimes words are included which are not used by the patterns, but just to make the example clear.

Example

under 23 men's [discus throw]

the tokens “under” and “men’s” are used as a context in the rule.  
[discus throw] is included to give the idea where in the text such pattern can occur.

### Independent rules:

21st birthday, 25th birthday, thirtieth birthday  
20-year-old, 20 years old, 20 years of age, 20yrs, aged 20  
 he/who became 9, turned 12

### Dependent rules:

under 23 men’s [discus throw] (uses Gender NE)  
 Tim Mack, 31, ... Anderson (33) (uses PersonName)

This grammar uses a Gazetteer list of ordinals such as “first, second... ninety-ninth”.  
 I believe that these rules are mostly domain-independent and thus can be reused by  
 other applications. Therefore this grammar is included in the appendix.

### Performance

This grammar recognizes the performance of an athlete during a specific sports trial in  
 a specific athletics competition. It needs to be processed after the Orthomatcher  
 because it depends on the entity PersonName for disambiguation.

All athletic performances in the documents are either length measures (jumping and  
 throwing events) or time measures (running and walking). Here are some examples:

*1:29:03.6*  
*1hr, 29min, 3sec*  
*one hour, twenty-nine minutes and three seconds*  
*2.34 [length in meters]*  
*19-00.25 [length in feet an inches]*

As it can be seen, most candidates for “Performance” can be expressed as a  
 combination of the following patterns:

```
one_two_digit: {1..99}
Float_number: {0.00 ... 99.99}
Time_number: {0:00... 59:59}
Number_word: {one... ninety-nine}
length_unit: {m, meter, metre, ... foot}
time_unit: {s, sec, ...hour}
```

However, in many cases this would result in false positives, for example:

*100m running.*  
*8:30 am; 8:30(GMT)*  
*5.01.2004 is not "5.01m"*  
*Heat 1: 1. Smith*  
*w:0.7*

1.4 m/s  
 headwind of 1.3; tailwind of 0.5m/s  
 over 2.00; under 3 seconds  
 0.23 outside his season's best  
 twenty-two seconds clear

Therefore, we specify that the rules should not match in the following cases:

- overlap with SportsName, Date, RoundName or Wind,
- comparison like "0.23s faster than"
- negative (e.g. -0.23 seconds)

The only exception is the pattern

PersonName – Performance, e.g. in the sentence:  
 "Bungei – 1:43.22 - Feofanova – 4.80 – in Madrid"

(This is why we need to process this grammar after the grammar of "PersonName").

## Date

Although the standard Gate installation provides rules for Date and Time expressions, I found that they perform poorly when applied to the athletics domain. Especially they produce many conflicts with the NE "Performance". Therefore I wrote them from scratch.

Here are some typical patterns which the rules recognize. Each line represents a rule. The comments are written in brackets.

18.04.04; 18 04 2004; [but not 18 04 04]  
 18/04/04; 18/04/2004;  
 18-04-04; 18-04-2004;  
 Monday, 15 May; Tuesday (16 May); Friday, 20th of May; [starts with day of week]  
 14-16 March; 21,22 Sep; 4 to 5 July; [date ranges]  
 Friday June 7; March 14-16; Sep 21,22; Sep 21/22; [English notation]  
 2003; 2004; [gazetteer list of years between 1900...2020]

These patterns are recognized without using context or other NE. Date is used to prevent spurious "Performance". (5.01.2004 is not "5.01m")

## SportsNameSubclasses

Subclasses of "SportsName" are recognized using a gazetteer list created in the training mode (see section 4.3.2).

## SportsName

This phase recognizes names of sports, such as "race walking 5km", "road running", "3000m steeplechase" using a gazetteer list, if it hasn't been already annotated in the phase "SportsNameSubclasses".

For example, "pole vault" will only be annotated as PoleVaultName, even if it occurs in the list SportsName.lst.

The named entities “Gender” and “City” are required as input, which means that this phase has to be processed after Coreference resolution.

## Ranking

This grammar recognizes the ranking of an athlete during a specific sports trial in a specific athletics competition. It also depends on the entity PersonName for disambiguation and therefore needs to be processed after the Orthomatcher.

Athletics Rankings can be expressed by numbers (most often in tables), ordinals, medal types or domain-specific verbs. Some examples with relevant context follow. The words that should be annotated as “Ranking” are underlined.

win gold  
1st place, first place  
silver medal  
 he was second with 1:44.96  
 the 29-year-old won the race  
 Smith, fifth at the Championships, returned to victory  
 X got the second place, Y was satisfied with third  
 B was in last  
 rankingVerb Ordinal  
1 . Annie Cahill , 65.33 ; 2 . Kelly Macintosh, 65.23  
5 . ( 24 . ) Martin Lel; 15 . ( - ) Mark Bett

For most of these examples a special rule was created.

rankingVerb is a gazetteer list of words such as  
 {back in; become; clinch; come; finish... settle for; win}

However, complete disambiguation against other NE is difficult by using only context.

*He was 2 seconds behind in fifth round, but still finished 1 second ahead in last.*

This sentence matches the patterns “be in fifth”, “finish second” and “finish last”. However, none of them refers to RoundName.

A solution could be to syntactically parse sentences to find out, to which noun the Ordinal refers.

The recognition of this NE is  
 - sentence in natural language  
 - tables

because most tables have predefined format, recognition from these tables is straightforward.

For example,

Men's javelin throw: 1. Jan Zelezny (Cze) 85.44 m; 2. Andrus Varnik (Est) 85.17; 3. Boris Henry (Ger) 84.74;

We see that the format is  
 Ranking dot PersonName Country Performance semicolon.

Therefore, table-based recognition is robust.

In the athletics news most information is provided in form of tables.  
 (This is also a reason, why shallow text processing was chosen)

### **SportsEventName**

This grammar recognizes names of a sports events, such as the “Olympic Games”.

After careful analysis of the manual annotations, it was discovered that most sports events include the following patterns, which can be entered in gazetteer lists or be expressed as Jape macros:

Event = {games; championship; cup; grand prix; meeting; festival; competition; event; sp; gp; invitational; outdoor(s); indoor(s); league...}

Loc = {world; international; european; ... iaaf; ncaa;... [City]; [Country]; [Nationality]}

Age = {u16.. u23; under 16; ...u23s; youth; junior; ...}

Year = {1900... 2020}

To remain specific enough and to prevent false positives, we demand that each sports event includes at least two of the above patterns. The following combinations are very common and were added as parts of rules:

*Ordinal – Loc – Event*

*Year – Loc – Event*

*Year – “Olympics”*

*Loc – Event – Age – Sport*

*Sport – Age – Event*

*Loc – Event – Year*

Some of the typical descriptors recognized by these rules include:

*10th IAAF World Championships*

*2000 World Half Marathon Grand Prix*

*2004 Olympics*

*world championships (under 23) in pole vault*

*german u-23 championships*

*discus throw u16s*

*rome golden league meeting*

*european cup of combined events first league*

*european junior 1985*

*world discus*

*Olympic*

As can be seen from these examples, this Named Entity has high potential for ambiguity. For instance, the string “Hamburg Marathon” could be parsed as (City + SportsName) or just SportsEventName.

In fact, many human annotations of this NE are inconsistent, which will be seen in the Performance Evaluation section.

To reduce the number of overlapping (and therefore spurious) annotations, the following strategy was used:

- Create a gazetteer list of all SportsEvents in the training set, remove all ambiguous entries.
- Match input text against entries of the gazetteer list.
- If a part of text matches an entry in the list, remove all annotations within its span.

In the above example, if the string “Hamburg marathon” is added to the gazetteer, all its occurrences will be recognized correctly and all overlapping annotations (“Hamburg” and “Marathon”) will be deleted.

Another strategy is based on the fact that many facts in the athletics news are presented in form of tables, for example:

SportsEventName	Ranking	Round	Performance	Wind	City	Date
7th IAAF World Championships in Athletics	1	f	14.88	-0.4	Paris	04 04 2004
28th European Indoor Athletics Championships	5	q	16.99	(1)	Madrid	04 03 2005

Therefore, by recognizing the format of these tables, the performance of the system can be improved.

Unfortunately, the html tags used to create the tables are not reliable to recognize the format, and some tables are created just by using non-breaking spaces.

Therefore, several table-based rules were developed, each tailored for a specific table format. The pattern which recognizes the table presented above is:

```
(
  (YEAR)?
  ((ORDINAL)? ({Token})[1,5] ):event //7th IAAF World..
  (ONE_DIGIT | TWO_DIGIT):rank // 1
  ({Token.root=="f"}|{Token.root=="q"}):round // f
  ({Performance}) // 14.88
  ({Wind})? //Wind
  (({Token.category==NPN})[1,2]):city // Paris
  (TWO_DIGIT TWO_DIGIT FOUR_DIGIT):date //04 04 2004
)
```

### SportsRoundName

The most common descriptor for this NE “final”, which accounts for 43% of all descriptors in the training set (290 of 667).

However, in many cases this word causes false positives, for example “final decision”.

Other recognized descriptors are e.g.:

“Round 2”, “fifth round”, “heat five”, “B-race”, “A 100 meters race”, “first semi-final”.

The first rule tries to identify entries directly from the table of sports events, which cannot be obtained by sentence analysis.

The advantage of this method is that if the table type is identified correctly, then the first of all, tables should be processed.

they are unambiguous, and require no sentence processing.

Improvement: automatically recognize table heading, then table-based extraction is robust to permutation of columns.

This method also deletes potential spurious matches.

It assumes the Table to have the following column names

SportsEventName	Ranking	SportsRoundName	Performance	Wind	City
-----------------	---------	-----------------	-------------	------	------

So, if this pattern matches, then all including Annotations can be created in this one rule. (E.g. `SportsEventName`, which is otherwise difficult to detect)

### **clean**

This phase removes any temporary annotations that have been created.

## **4.2.7 Relation Extraction**

In this phase we want to identify surface-level relations that hold between the extracted Named Entities.

According to [17], because we are dealing with *meaningful* texts in natural language, we can assume that words are put together not randomly, and thus, two NE occurring within the same sentence are always related.

Therefore, the most general algorithm (implemented in [17]) takes each pair of NE in a sentence and adds “associatedWith” relation between them. As the author points out, this approach tends to over generalize and can lead to erroneous results in the reasoning phase.

My extensions of this algorithm are as follows:

### **Use the ontology to decide which relations are feasible**

According to the AEO ontology design, all relation names are derived directly from the types of NE that can engage in that relation. For example, the only specific relation between `PersonName` and `Country` is “`personNameToCountry`”.

On the other hand, neither “`ageToCity`” nor “`cityToAge`” relation is defined in the ontology, which means that relation between “Age” and “City” is not useful for template relation extraction.

### Prioritize proximity

Based on intuition and statistical analyses, we can assume that the closer in the text NE, the higher is the probability that they are related.

This is necessary for disambiguation.

Consider this example:

*"Bungei bettered Kipketer's previous mark of 1:43.88"*

The relation extraction phase "sees" the following input:

PersonName<sub>1</sub> PersonName<sub>2</sub> Performance<sub>1</sub>

As we know, a Performance can be related with only one PersonName.

To choose the right candidate, we use distance between the NE as a measure and conclude that PersonName<sub>2</sub> and Performance<sub>1</sub> should be related.

Of course, one can find examples where this approach yields wrong results; nevertheless, in most cases this approximation works well.

### Restrict cardinality of relations

Based on intuition we may demand that each NE is allowed to be assigned to only one NE of a specific type. For example, a person can have only one nationality, age, gender etc.

Therefore, before adding a relation between 2 NE it is first checked whether each of them is not "already taken".

A few exceptions exist to this rule. For example, multiple Persons can take part in "Olympic Games", "Semi-Final", "Running 800m".

These exceptions are added to a list:

```
String [] cand = {curClass, nextClass}; //current pair of NE types

String[][] multiple = {
    {"SportsEventName", "PersonName"}, // Persons in "Olympic Games"
    {"SportsRoundName", "PersonName"}, // Persons in "Semi-Final"
    {"SportsName", "PersonName"},      // Persons in "Running 800m"
    {"SportsEventName", "SportsName"}  // Sports in "Olympic Games"
};
for (int a=0; a<multiple.length; a++) {
    if (Arrays.equals(allowed[a], cand) ) {
        exceptionAllowed = true;        // allow current pair to form a
relation
        Out.println("multiple relations allowed for "+curClass+",
"+nextClass);
    }
}
```



## The Algorithm

The simplified version of the algorithm looks as follows:

```

put all NE in an array,
sort by position in the document.

For each NE n
  For each NE m
    if (rel(n, m) or rel(m,n) is valid){
      if (free(n) && free(m) && nearest(m,n) || multiple(n,m) ) {
        add rel(n,m);
      }
      else
        continue with next m.
    }
  end
end
end

```

## Example

Going back to our running example:

*Bungei wins men's 800m in style.*

*London, July 26th - Kenya's 2004 Olympics silver medalist Wilfred Bungei (24yrs) set a new world record with a 1:43.72 clocking to better Denmark's Wilson Kipketer previous mark of 1:43.88 in the semi-finals held tonight at the Wembley Stadium.*

The system log for the first sentence is as follows:

```

.. Bungei .... wins ?
Bungei ---- personNameToRanking ---- wins
.. Bungei .... men ?
Bungei ---- personNameToGender ---- men
.. Bungei .... 800m ?
>> multiple relations allowed for PersonName, SportsName
800m ---- sportsNameToPersonName ---- Bungei
.. wins .... men ?
genderToRanking is an invalid relation
.. wins .... 800m ?
800m ---- sportsNameToRanking ---- wins
.. men .... 800m ?
sportsNameToGender is an invalid relation

```

The extracted relations and Named entities can be visualized using the Jambalaya plugin of Protégé:

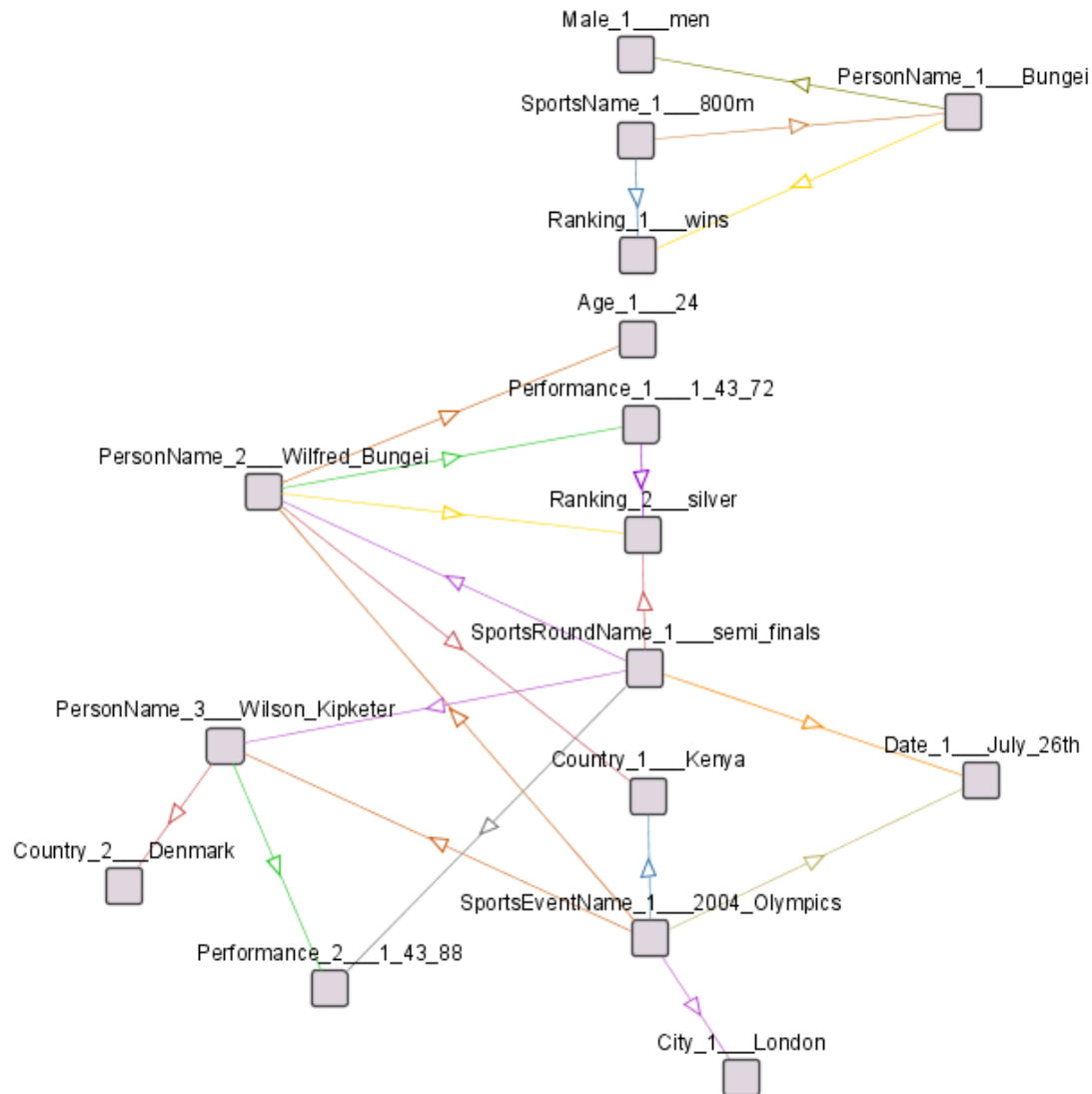


Figure 9: Relations between NE extracted from the running example

This graph has been automatically generated from the ABox produced by the developed System.

### extract relations during NE recognition

An extension of the algorithm would be to perform relation extraction and NE recognition simultaneously.

Most of the developed NE extraction rules use other NE as context, assuming that they are related, for example:

```
// Przemko Meier of Poland
// Dshchuy Miller of the United States
// Xyphjer Smith from Jamaica
// Wjatscheslaw Anderson ( AUS )
```

```
Rule: NationalityContext2
(
  (
    ({Token.orth=="upperInitial"} LAST_NAME)
  ):cur
  (OF|FROM|OPEN_BRACKET) (THE)?
  {Country}
)
-->
:cur.PersonName = {}
```

This rule uses Country as a context to recognize a PersonName. The Country's ID could be added as a feature to PersonName annotation, which could be then used in the relation extraction phase.

### **relSameAs**

In this phase we utilize information provided by the Orthomatcher to add identity relations between NEs having matching names. The corresponding Ontology relation is owl:sameAs, which makes identical NEs share all their properties and relations.

As described in [29], the abductive retrieval inference service has the task of computing what should be added to the Knowledge Base in order to answer the query with true. By merging together the information of the matching NEs, less assumptions in the abduction phase will have to be made.

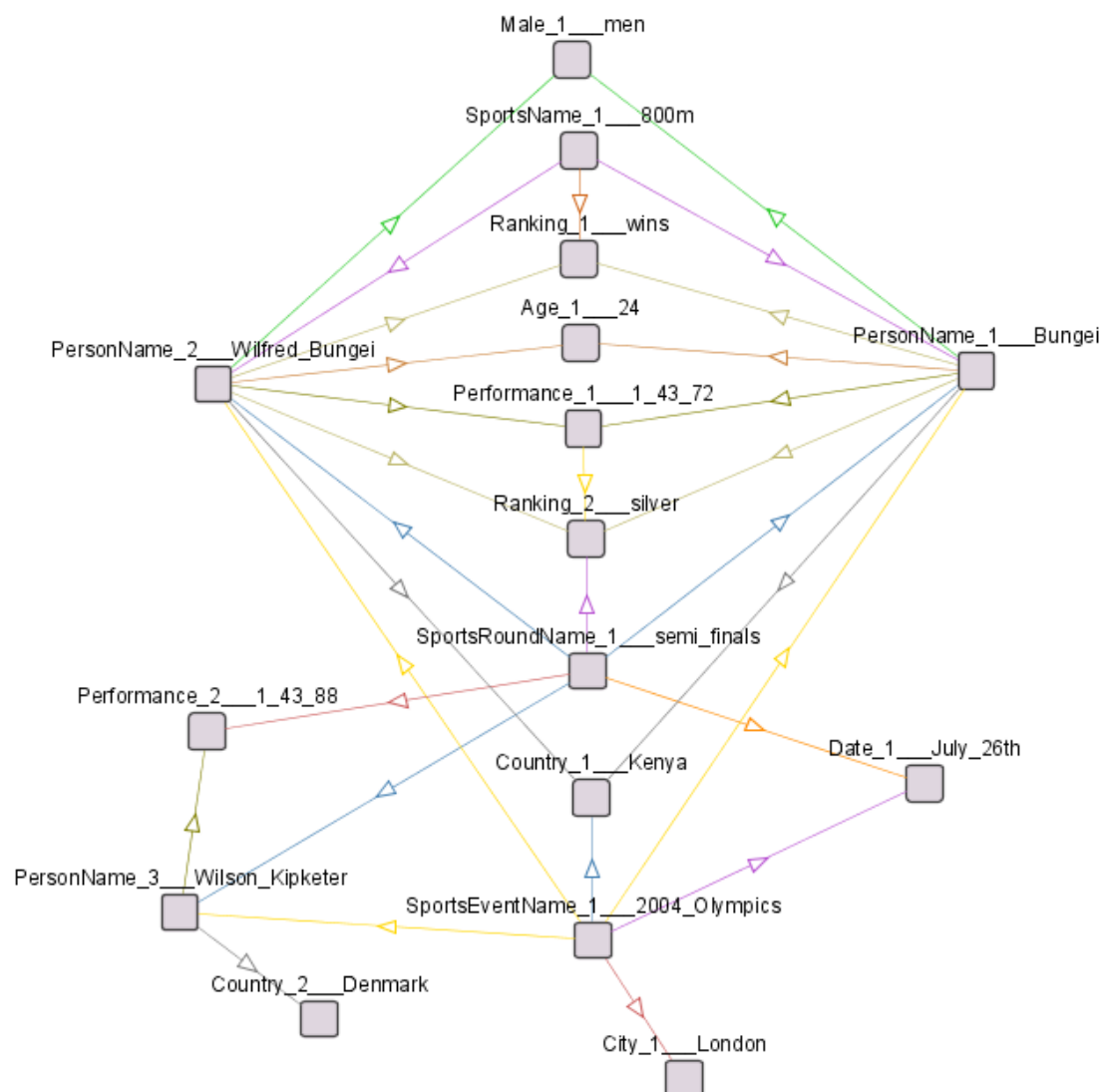


Figure 10: Named Entities and relations between them after adding owl:sameAs relation between PersonName1 and PersonName2

## 4.3 Training Mode

### 4.3.1 The need for adaptability

As pointed out in [31], the world is changing at a fast pace, and new sports names, person names emerge every day. Therefore, the system should be able to be updated efficiently when new manually annotated data becomes available.

The purpose of the training mode is also to make the system adaptable to changes in the ontology structure, and to learn new descriptors for Named Entities (NE).

Here are the processing resources that accomplish this:















!	Name	Type
	 remove annotations	Document Reset PR
	 processOrigMarkups	Jape Transducer
	 alternateTokeniser	ANNIE English Tokeniser
	 single-NewLine splitter	ANNIE Sentence Splitter
	 createTestAnnots	Jape Transducer
	 gazettTest	ANNIE Gazetteer
	 createTestDocs	Gazetteer List Collector

Figure 11: Processing resources used in the training mode

The training mode accomplishes the following tasks provided manually annotated content:

- 1) create a gazetteer list of ontology concepts
- 2) collect lists of PersonNames, Cities and Countries from manual annotations
- 3) create test documents which contain the collected NE.

### 4.3.2 Collect ontology concept names

I found that the number of subclasses of the MLC “SportsName” often varies among different versions of the ontology. For example, some versions have “Running800m” modeled as a subclass, which means that when the string “running 800m” occurs in an input document, it should be annotated as “Running800m”, and not as “SportsName”.

To adapt to such ontology changes, a list of subclasses of “SportsName” is collected. For example, if one of the subclasses is PoleVaultName, the entries “pole vault”, “POLE VAULT” and “Pole Vault” will be added to the gazetteer list. This list is then used in the NE phase “dynamic list”.

### 4.3.3 Collect Gazetteer lists

The developed system heavily relies on gazetteer lists for recognition of such NE as “PersonName” and “SportsEventName”. Therefore, it is essential that the gazetteer lists are updated regularly to make sure that the performance does not deteriorate over time.

In the current version of the software, the Gazetteer lists were collected from 300 manually annotated documents. The print versions of 29 of them were used in the Performance evaluation. Their names can be found in Appendix A.1.1.

### 4.3.4 Create Test Documents

#### Motivation

We need to test the NE extraction process, by evaluating the quality of NE detection for each NE.

To get reliable statistics, we need at least 100 NEs.

However, some NE (e.g. "Stadium") are rare, and 10-30 documents would need to be processed to get 100 of them.

The problems are:

- slow processing
- can't visualize results efficiently (we see only 1 document in AnnotDiff tool)

### Idea

collect text parts from different documents that contain the required named entity.

The purpose of this Phase is to run over a corpus once and create a test document:

- each sentence contains at least one NE of the specified type.

The advantages are:

- the document is short:
  - fast processing
  - can visualize annotations efficiently (we see all training data at once)
- can apply AnnotDiff tool and evaluate Recall instantly (each "line" in the test file must contain an annotation created by our system)

Shortcoming: cannot test Precision this way. (Can't measure how many false positives the system creates on a "usual" text)

The test documents simply take a 7-Token context of the given entry, within a sentence. (Because the grammars of the Analysis mode look for NE only within a single sentence). Note: the described idea is similar to Annic.

I implemented this phase because:

- it allows to see all matches or misses at once
- very fast (only 1 test doc per NE)

However, as will be explained later, this method is not enough to test the performance of the system, because it is impossible to test how many spurious annotations the system produces.

(The generated docs only allow to test recall, i.e. how many of the correct NE were found)

Note: testDocs are not a prerequisite for the analysis phase and can be safely turned off. I used them only to quickly test the Recall of the analysis phase.

### Testing phase

In the first test we only check recall

(Annic allows you to test one pattern at a time)

AnnotDiff tool only visualizes one document. Therefore such rare entries as "stadium" that occur at most 1 time per document cannot be tested efficiently.

The corpus benchmark tool ("Verbose mode") only displays the spurious or missing annotations, but doesn't show the context.

The procedure to test is:

Run the pipeline. It will generate the test documents. (If the human annotations change, need to manually delete the content of each list).

So, we have the documents for each interesting NE.

Then: populate corpus with (only) these lists.

→ instead of testing 300 documents we only need 1.

Note: this approach was designed for rare NE, such as “Stadium”. Applying it to such things as “PersonName” can actually increase the size of the test document (because for each NE in the manual annotations, 8 Tokens of context are added).

## Implementation

### ***document reset***

- remove all annotations created by previous runs of the pipeline

### ***processOrigMarkups***

- extracts manual annotations from the text. As mentioned earlier, the input are html documents, with manual annotations save in Tags named “span”. These manual annotations (in Ellogon format) are converted to Gate format.

for example:

```
span = {class = keyword.City, id=City9366, onClick=KonText();}  
City = {}  
ONTO = {class = City}
```

### ***alternateTokeniser***

splits text into tokens. Hyphens are separate tokens, words and numbers are separate as well (like u23s.)

This allows more flexibility when writing rules for the analysis phase.

We need the tokeniser because need token context for generation of test docs.

### ***sentence Splitter***

at each new line the sentence is split.

We need sentences for the training mode, because we want to generate test Docs. (explained later).

All our grammars recognize NE only within a single sentence, therefore the test data should be single sentences.

### ***gazetTest***

This PR needn't be part of the pipeline, but must be loaded into gate. (it is put in the pipeline here only for visualization)

Gazetteer list collector needs a Gazetteer PR instance to operate.

### ***createTestDocs***

Is a Gazetteer list collector.

Specify in it's parameter list, which annotation types should be collected, and it will harvest them.

## Example 1

Suppose, the following text was annotated by human annotators

Friday, 12 August 2005 - Women Pole Vault Final  
Yelena Isinbayeva (RUS), Monika Pyrek (POL) and Pavla Hamackova (CZE)  
took part in the World Championships.

date1: Friday, 12 August 2005  
 Gender1: Women  
 SportsName1: Pole Vault  
 RoundName1: Final  
 PersonName1: Yelena Isinbayeva  
 County1: RUS  
 PersonName2: Monika Pyrek  
 County2: POL  
 PersonName3: Pavla Hamackova  
 County3: CZE

Assume, we want to train the system to recognize the named entity "Country". We specify "Country" as input for the Gazetteer list collector. For the contextSize = 2, the following documents will be created:

Country.lst	CountryTest.lst
RUS	Isinbayeva ( RUS ) ,
POL	Pyrek ( POL ) and
CZE	Hamackova ( CZE ) took

## Example 2

We want to create a testDoc for "Stadium":

1. load the pipeline "gazetteerListCollector\_createTestDocs.gapp"
2. create "StadiumSentence.lst" in the gazetteer directory (C:\GATE\plugins\\_BOEMIE\gazetteer)
3. open listsToPopulate.def, insert entry "StadiumSentence.lst:StadiumSentence:inferred"
4. set "AnnotationTypes" of gazetteer list collector to "StadiumSentence"
5. reinitialize gazetteer and collector (FSM will be updated)
6. populate corpus with documents containing Original markups such as `<span={class=keyword.City,id=City9366,onClick=KonText();}/>`
7. run the pipeline



Stockholm's intimately charming Olympic Stadium. finish line in Athens' Marble stadium as the new Athletics Gala at the Grimaldi Forum, Monte-Carlo Helsinki's newly renovated Olympic stadium. chilly night in the Olympic stadium, the 22 new surface in the Olympic stadium which had earlier Elsewhere tonight in the Olympic stadium, Derek Miles, 000 in the Stade Roi Baudouin. suddenly descended over the Stade Roi Baudouin. which was the historic Olympic Village from the 1936 Games arrive at the "Pino Dordoni" stadium, which on the brand new Pontaise track exactly ten years of Fredericks at the Pontaise Stadium in Lausanne. - in Monaco's Stade Louis II. London - Crystal Palace stadium in the South Palace Welcomes Royalty of Athletics) entered Brussels' King Baudouin Stadium tonight in contention place in Birmingham's National Indoor Arena (14-16 competing in London's Crystal Palace evening at the 1952 Olympic stadium. This evening in the Sportarena Yelena Isinbayeva set a pleased to win as Crystal Palace is the arena where London Grand Prix at Crystal Palace, including three sub Grand Prix - in Linz Main Stadium. night in the 1912 Olympic stadium, which was who trains in the Stade Louis II, completed an unbeaten

Set	Start	End	Id	Features
	33	40	4821	{class=Stadium, rule=StadiumWordPre}
	33	40	4820	{class=Stadium, rule=StadiumWordPre}
	66	81	4823	{class=Stadium, rule=StadiumWordPre}
	66	81	4822	{class=Stadium, rule=StadiumWordPre}
	124	138	4824	{class=Stadium, rule=StadiumGaz}
	182	189	4825	{class=Stadium, rule=StadiumWordPre}

Figure 12: An automatically generated test document for the named entity "Stadium"

The advantage of this method is also that it allows to identify conflicting rules. The more Annotations of the same type match the same part of text, the darker is the color. For example, in the first line the word "Olympic" was matched 2 times, whereas "Grimaldi Forum" only once.

The Gate annotation list also shows which grammar rules created the conflict.

By design, each line in the document is supposed to have at least one NE of the given type. Therefore, whenever a line is not marked, it indicates a miss. (In the figure above, it is the line "Palace Welcomes Royalty of Athletics").

# 5 Performance evaluation

## 5.1 Test setup

Due to time limitations only the NE recognition step was evaluated.

The performance was evaluated on 30 documents from the training set and 30 previously unseen documents (evaluation set).

Most provided html documents contain headers and hyperlinks, which were not annotated by humans. If they were used for evaluation, this would result in many “false positives”:

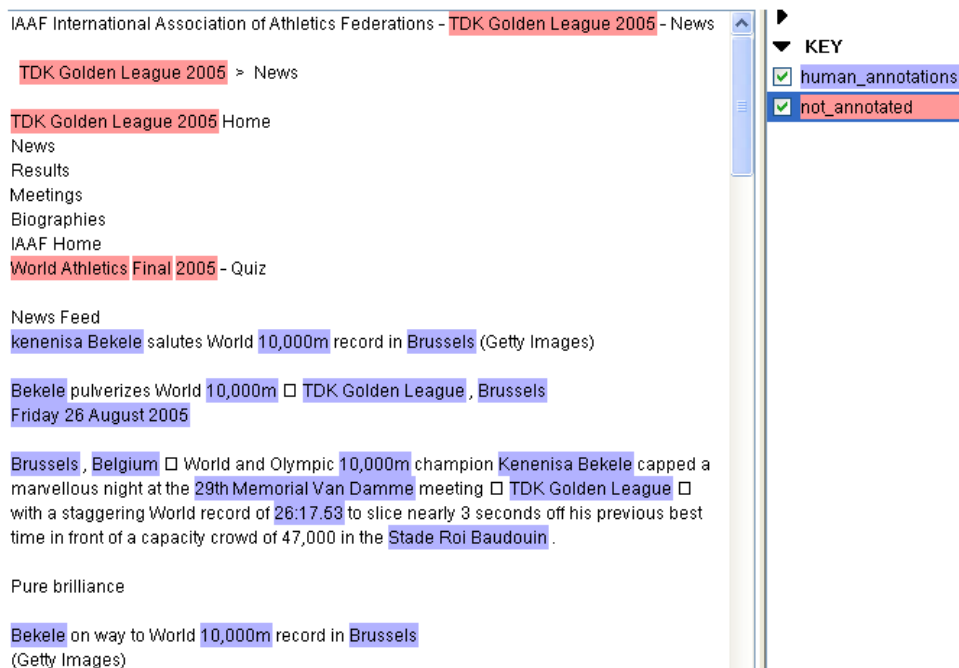


Figure 13: Headers and hyperlinks were not annotated by humans, which results in “false positives” (red)

A solution to this could be to examine the html tags and filter out those parts of text corresponding to irrelevant information. However, I found that html tags alone are not sufficient to decide what to filter out.

Some approaches use vision-based segmentation algorithms [12]. However, page segmentation is outside the scope of this thesis.

Therefore, for evaluation purposes only those documents were taken, that didn't contain headers, that is, the available "print versions" of webpages. Unfortunately, there was only 60 of them available. The list of documents used for evaluation is located in the Appendix.

## 5.2 Evaluation metrics

### 5.2.1 Definitions

Key Set: Set of human-marked annotations (ground truth).

Response Set: System-marked annotations.

#### **Coextensive**

Two annotations are coextensive if they hit the same span of text in a document. Basically, both their start and end offsets are equal.

#### **Overlaps**

Two annotations overlap if they share a common span of text.

#### **Correct**

Annotation A of the response set is correct, if there exists an annotation B from the key set, such that:

- coextensive(A, B)
- type(A) == type(B)

#### **Partially correct**

Annotation A of the response set is correct, if there exists an annotation B from the key set, such that:

- overlaps(A, B)
- type(A) == type(B)

Partially correct responses are normally allocated a half weight.

#### **Missing**

This applies only to the key annotations. A key annotation is missing if either it is not coextensive or overlapping, or if one or more features are not included in the response annotation.

#### **Spurious**

This applies only to the response annotations. A response annotation is spurious if either it is not coextensive or overlapping, or if one or more features from the key are not included in the response annotation.

The metrics used for evaluation are defined as follows:

$$Precision = \frac{Correct + \frac{1}{2} Partial}{Correct + Spurious + Partial}$$

$$Recall = \frac{Correct + \frac{1}{2} Partial}{Correct + Missing + Partial}$$

$$F_1 - Measure = 2 \frac{Precision * Recall}{Precision + Recall}$$

Finally, the Overall average measures are obtained by averaging over the number of documents  $N$  and annotation types  $M$ :

$$F_{average} = \frac{1}{N * M} \sum_{i=1}^N \sum_{j=1}^M F_{i,j}$$

## 5.2.2 Corpus Benchmark Tool

The evaluation was carried out using the Corpus Benchmark Tool of GATE. This tool enables two versions of an annotated corpus to be compared, i.e. the annotations produced by the System, and the Human annotations. For each annotation type, Precision, Recall, F-measure are calculated. The output of the tool is written as a table in an HTML file.

## 5.3 Results

### 5.3.1 Training set

Annotation Type	Correct	Partially Correct	Missing	Spurious	Precision	Recall	F-Measure
PersonName	1791	20	39	14	0.9868	0.9735	0.9801
Gender	263	0	0	21	0.9261	1.0000	0.9616
Age	61	3	7	1	0.9615	0.8803	0.9191
Stadium	20	1	0	0	0.9762	0.9762	0.9762
Performance	1018	15	29	47	0.9495	0.9656	0.9575
Country	942	4	34	64	0.9347	0.9633	0.9487
City	423	1	36	35	0.9227	0.9207	0.9217
Date	208	3	45	18	0.9148	0.8184	0.8639
SportsName	347	3	21	172	0.6676	0.9394	0.7805
Ranking	754	0	215	97	0.8860	0.7781	0.8286
SportsRoundName	50	1	9	32	0.6084	0.8417	0.7063
SportsEventName	374	44	14	169	0.6746	0.9167	0.7772

Table 2: Performance achieved on the training set

Overall average precision: 0.8651997841810992  
 Overall average recall: 0.9184930665651431  
 Overall average fMeasure : 0.8654693482981329

### 5.3.2 Evaluation set

Annotation Type	Correct	Partially Correct	Missing	Spurious	Precision	Recall	F-Measure
PersonName	2127	79	153	41	0.9642	0.9184	0.9407
Gender	355	0	0	17	0.9543	1.0000	0.9766
Age	88	4	5	7	0.9091	0.9278	0.9184
Stadium	16	1	7	1	0.9167	0.6875	0.7857
Performance	1066	16	31	57	0.9429	0.9650	0.9538
Country	1290	4	59	99	0.9275	0.9549	0.9410
City	452	1	85	28	0.9407	0.8411	0.8881
Date	179	5	40	10	0.9356	0.8103	0.8684
SportsName	523	4	22	259	0.6679	0.9563	0.7865
Ranking	1279	0	333	92	0.9329	0.7934	0.8575
SportsRoundName	31	0	29	40	0.4366	0.5167	0.4733
SportsEventName	331	67	16	196	0.6136	0.8804	0.7232

Table 3: Performance achieved on the evaluation set

Overall average precision: 0.8661736589539282  
 Overall average recall: 0.8749211749301447  
 Overall average fMeasure : 0.8426918176165006

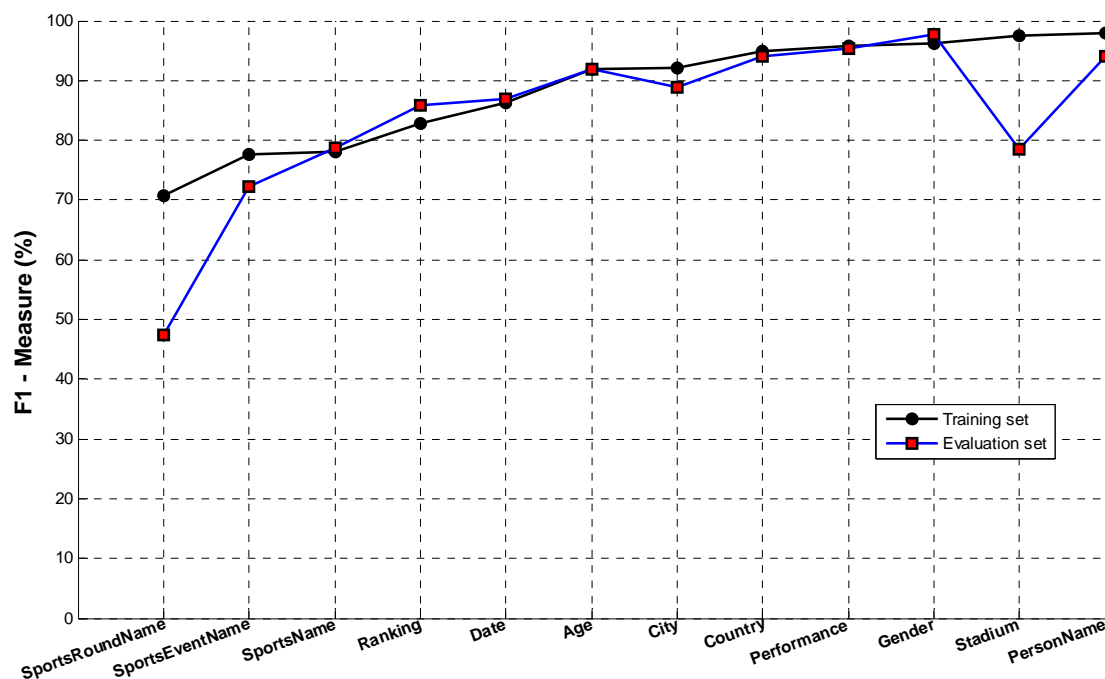


Figure 14: Average F1-measure for each NE type in the training set and evaluation set

The per-document results produced by the corpus benchmark tool are located in the appendix.

### 5.3.3 Processing speed

#### Test Machine:

Windows XP SP 3  
2GB RAM, 2 x 2.4 GHz

Java: JVM 1.6.0\_10-rc-b28  
GATE 5.0, build 3099

#### NE recognition task:

Input: 30 documents in html format.  
Total size: 0.98MB  
Processing time: 15.59s  
→ 64,37Kb/second

#### Complete functionality:

(NE recognition, ontology population, relation extraction, export to file)

Input: 30 documents in html format.  
Total size: 0.98MB

Processing time = 15 minutes  
 → 1Kb/second

Observation: The more data the ontology contains, the slower the operations become.

Here are the individual processing times, assuming that the ontology is empty at start:

Document	Size	# NE	# relations	proc. time
ID31565	10kb	28	15	1.7s
ID21554	15kb	91	102	2.6s
ID27365	23kb	157	155	4.1s
ID31771	34kb	215	474	5.7s
ID30642	45kb	350	1042	9.3s
ID26288	61kb	495	1769	13.5s
<b>Sum</b>	188kb	1336	3557	36.9s

Table 4: Processing times when ontology is cleaned before processing each document

To test how the number of elements in the ontology influences the processing speed, the documents were processed after each other without cleaning the ontology after each document.

Run #	processing time
1	37.5s
2	61.7 s
3	90.4 s
4	119.9s
5	150.5s

Table 5: Processing times without cleaning the ontology

The table shows that the processing time is linearly proportional to the number of instances and relations present in the ontology.

## 5.4 Interpretation of results

### 5.4.1 Named entity recognition

The figures show that the recognition of named entities whose grammars rely on gazetteer lists mostly becomes worse on the evaluation set.

For example, the F-measure on the NE “Stadium” drops from 97% to 78%.

On the other hand, NE defined using only regular expressions have the F-measure remaining constant (Date, Age, Performance).

The performance on the NE “SportsRoundName” drops from 70% to 47%. The possible reason is that some documents in the evaluation corpus use a previously unknown format or descriptors for this named entity.

Please note that most gazetteer lists used for the NE task were obtained automatically by collecting manual annotations from the training corpus.

As can be seen from the table, the named entities “Stadium” and “City” exhibit high precision and relatively low recall, which means that many descriptors are missing in the gazetteer lists.

I believe that by populating those lists with more entries could significantly improve the performance on the aforementioned NEs. This was not done in order to see how well the system “learns”.

### Inconsistent annotations

Another issue are inconsistent annotations made by humans. Especially the NE “SportsEventName” is not clearly defined.

For example, the training corpus of 300 documents contains this entity 3306 times and in 244 cases the descriptor is “Olympic”. The most common context is “champion”, “medal” or “SportsName”.

However, in about as many cases this rule does not apply, which leads to “false positives”.

(Still, this rule was included in the grammar, because it yields higher overall average F-measure).

Berlin (Getty Images) Olympic champion beats World gold medallist one would expect of an Olympic champion, his 87. the nine woman field. Olympic champion Natalya Sadova , who Tatyana Lebedeva duly won the TDK Golden League Jackpot of One Mill Berlin , the final TDK Golden League meeting of the summer. Tatyana week to go until the World Athletics Final in Monaco ( 9 / at this afternoon's ' ISTAF Berlin ', the final TDK Bahraini will contest his other World Championship title distance, the 150 hopes for their World and Olympic finalist Tobias Unger , who 1500m , at the World Athletics Final next weekend in Monaco ( generation the way, with TDK Golden League wins in Rome , Zürich now can go to the World Athletics Final (9 / 10) s Yamile Aldama , the World Indoor silver medallist, showed she series of top-8 TDK Golden League results this season (5 , who in turn beat Olympic champion Stefan Holm ( SWE 44 territory in Berlin , Olympic silver medallist Mbulaeni Mulaudzi was his dismal display at the World Championships in Helsinki last month wh all six of the lucrative TDK Golden League meetings □ had to be satisfier be satisfied with second behind Olympic silver medallist Chrisopiya Deve easy, however, as Olympic gold medallist Andreas Thorkildsen was victory for the World and Olympic champion but she was unable venue, having won the World Athletics Final her in 2003 . Three . Among others, double Olympic silver medallist Terrence Trammell will

- Sentence
- SpaceToken
- Split
- SportsEventName
- SportsName
- SportsRoundName
- Spurious
- Stadium
- Token
- Unknown

Figure 15: “Olympic” is a very common descriptor of the NE “SportsEventName”. However, human annotations are highly inconsistent here, which leads to many false positives.

Another problem is that such strings as "Berlin marathon" are sometimes annotated as "City + SportsName" and sometimes as "SportsEventName".

One error in this case produces three -- one miss and two false positives.



Inconsistent annotations are also a reason why such gazetteer-based NE as “Country” do not achieve the F-measure of 100%. Some nationalities (“German”, “French” etc) were marked as “Country”, because the BOEMIE annotation tool [12] doesn’t differentiate between “Country” and “Nationality”.

However, if “Nationality” descriptors are added to the gazetteer list of “Country”, this will cause many false positives, in phrases like “German athlete”.

### 5.4.2 Relation Extraction

We have seen that processing time on this task is linearly proportional to the number of elements in the ontology.

One way to solve this problem is to store all information (URIs and relations between NE) in the annotations themselves instead of populating the ontology.

However, it is unclear how reasoning can be carried out if the ontology is not populated.

## 6 Discussion

### 6.1 Comparison to other projects

#### 6.1.1 NE recognition

The most recent project which also carries out IE on texts in the Athletics domain is BOEMIE [31]. According to [14], the BOEMIE named entity recognizer uses Conditional Random Fields and exhibits “precision of 90 %, and recall approaching 86 %”. However, the authors do not provide details on the test conditions (what NE on what texts were evaluated).

The corresponding values achieved by the developed system on the evaluation set are: precision: 86.6%, recall: 87.5%

This shows that hand-coded grammars have the potential to outperform machine learning systems.

#### 6.1.2 Relation extraction

The context-free grammar approach for relation extraction is presented in [14].

The important details are:

- The evaluation was limited to relations occurring within sentence boundaries,
- only relations involving names and properties related to athletes were considered,
- relations not fulfilling the above requirements were removed from the text corpus.

The relation extraction algorithm presented in this thesis has the following advantages:

- it is very generic and therefore domain-independent,
- doesn't require any training.

The only adjustable parameter is the list of multiple relations that should be allowed. (E.g. allow “SportsName” form multiple relations with “Performance”)

#### 6.1.3 Classification of the System

According to [1],

“The third level of ontological knowledge that can be exploited by IE systems concerns the concepts' properties and/or the relations between concepts. For example, the “athlete” concept can be defined by the “name” property filled by a string value, the “nationality” relation filled by values of the concept “country”, the property “age” filled by a numeric value, etc. These properties and relations guide the information extraction

system in various processing stages, from named entity recognition to template relation extraction, independently of the techniques used. In a rule-based approach, the knowledge engineer writes rules for detecting specific types of relations inside a text (e.g. “nationality” associates a person-name with a country-name, therefore the corresponding rule looks for instances of such named entities inside a sentence associated in different ways, such as “... the Russian T.Lebedeva...”, “T.Lebedeva, the Russian athlete”).

The developed system fits this description, therefore it is level 3.

### **Descriptive features of information extraction systems**

A framework for classification and comparison of Ontology-based IE systems is described in [1]:

1. Use of the ontology in the IE process, i.e. which level/levels of the ontological knowledge are exploited.
2. Initial requirements in terms of the type of input documents (raw texts, semi-structured such as web pages, fully structured content in the form of tables or databases) and the preprocessing required (lexical analysis, syntactic analysis, semantic analysis).
3. Ontology features, such as the knowledge representation formalism, the ontology implementation language, the reasoning mechanism employed, the ontology size.
4. Extraction process: rule-based, machine learning-based, hybrid.
5. Completion result (e.g. annotated corpus, filled templates, populated/enriched ontology)
6. Domain portability.

Using this definition, we have:

- 1) The system exploits conceptual properties and relations
- 2) Any input type is fine. The system can work with plain texts as well as XML or HTML. No preprocessing is required, because the system uses its own preprocessing components.
- 3) Ontology language: OWL. No reasoning is employed, because only surface-level information is extracted. No sentence parsing is performed. The Athletics Event Ontology (AEO) has 340 OClasses, 99 ObjectProperties and 22 DatatypeProperties.
- 4) The extraction process is rule-based. Gazetteer lists are collected automatically from training texts, which can be considered machine learning.
- 5) Completion result: annotated corpus, populated ontology, exported ABox.
- 6) Domain portability: at least 6 of the 12 named entities are domain independent. (Date, Age, City, Country, Gender, PersonName)

## **6.2 Robustness to changes in the Ontology**

Let's assume “Weight Lifting” is added as a new subclass of SportsName to the ontology. Now the values of “Performance” can be such Strings as “120kg”; “230pounds” and so on.

A machine learning system would have to

- get at least 100 documents with the new “Performance” manually annotated
- completely retrained to recognize this new type.

In contrast to that, the developed system will only need to be updated with just one Jape rule.

### **6.2.1 Portability to a new Domain**

I believe that several grammars developed for the 12 NE of the Athletics domain can be reused, as follows:

Date, Age, City, Country, PersonName, Gender are very common in almost all kinds of texts. Most of these grammars are context-free, therefore little or no manual adaptation of the rules is necessary.

Ranking is a very common NE, for example in politics, education system etc. It could be used to recognize ToDo lists or other ordered lists of items.

Several rules of “Performance” can be adapted to recognize time expressions and length measures.

The grammar for “Stadium” could be reused to recognize names of theaters, bookstores etc. One would need to replace the gazetteer list of Stadium names by a list of theater names etc.

Portability to another foreign language is also very straightforward. One would only need to replace the gazetteer lists with the lists for the new language.

# 7 Conclusions

To sum up, the developed rule-based IE system has the following advantages and disadvantages, compared to other investigated IE systems.

## 7.1 Advantages

- Very little manually annotated data is required to train the system. Most gazetteer lists (e.g. Countries, Cities, Person names, sports events) are available on the Web and could be reused.
- Rules can be developed solely by using intuition and background knowledge about the world, e.g. the grammars for "Age", "Date" and "Gender".
- Flexibility. The system can be easily adapted to changes in the ontology or NE definitions by adding new rules, which meet the new specifications. Since the JAPE language is very intuitive, the users of the system can add new rules to extend the system's functionality.

## 7.2 Disadvantages

- Domain expertise is required to develop some rules. For example, one must know exactly how "Sports Event name" is defined, and how it should be annotated.
- Sensitivity to noise and errors made by human annotators. If descriptors of named entities are automatically collected from the training data, errors made in the gold annotations tend to replicate. For example, if the string "Hamburg" was mistakenly annotated as a "Country", then all occurrences of this string in the evaluation corpus will also be annotated as "Country". To alleviate this problem, the gazetteer lists had to be manually inspected after each training phase, to remove erroneous annotations.

## 8 Future Work

- To overcome the noise sensitivity problem, the frequency of annotations could be considered during the NE extraction process. For example, if "Hamburg" is annotated as "City" 10 times, and as "Country" only one time, we can say that the "Country" annotation is erroneous. However, this is still a challenge if only a few training documents are available or if the named entity is rare, for example "Stadium".
- Techniques of principle component analysis and latent semantic analysis could be used to disambiguate based on context. For example, analyze which terms occur most frequently in the sentence with "RoundName" (co-occurrence matrix)
- Investigate combination of rule-based and machine learning based systems
- Carry out demarcation of news items by applying a vision-based page segmentation algorithm [12]
- Investigate Noun Phrase and Verb Phrase chunking or full parsing of sentences [35]
- Detect synonyms of NE using WordNet or OpenCyc [34]
- Evaluate the performance of relation extraction [21],[4]
- Develop an algorithm which would automatically generate Jape rules, or at least a good approximation. Since JAPE is a finite state transducer, I believe that some of the algorithms used for lexer or parser generation (Lex, Flex, Yacc) could be reused.

# A Appendix

## A.1 List of documents in the corpora

### A.1.1 Training set

newsId=21554,printer.html  
newsId=23132,printer.html  
newsId=26434,printer.html  
newsId=27365,printer.html  
newsId=26288,printer.html  
newsId=30642,printer.html  
newsId=31004,printer.html  
newsId=31276,printer.html  
newsId=31565,printer.html  
newsId=31877,printer.html  
newsId=32079,printer.html  
newsId=36054,printer.html  
newsId=36938,printer.html  
www\_iaaf\_org\_news\_newsId=21070,printer.html  
www\_iaaf\_org\_news\_newsId=25867,printer.html  
www\_iaaf\_org\_news\_newsId=27380,printer.html  
www\_iaaf\_org\_news\_newsId=30642,printer.html  
www\_iaaf\_org\_news\_newsId=31771,printer.html  
www\_iaaf\_org\_news\_newsId=31877,printer.html  
www\_iaaf\_org\_news\_newsId=31965,printer.html  
www\_iaaf\_org\_news\_newsId=32079,printer.html  
www\_iaaf\_org\_news\_newsId=35727,printer.html  
www\_iaaf\_org\_news\_newsId=35884,printer.html  
www\_iaaf\_org\_news\_newsId=36054,printer.html  
www\_iaaf\_org\_news\_newsId=36467,printer.html  
www\_iaaf\_org\_news\_newsId=36517,printer.html  
www\_iaaf\_org\_news\_newsId=36929,printer.html  
www\_iaaf\_org\_news\_newsId=36974,printer.html  
www\_iaaf\_org\_news\_newsId=37103,printer.html

### A.1.2 Evaluation set

newsId=22172,printer.html  
newsId=22233,printer.html  
newsId=27380,printer.html  
newsId=31278,printer.html  
newsId=31945,printer.html  
newsId=33708,printer.html  
newsId=35525,printer.html  
newsId=35884,printer.html

newsId=36080,printer.html  
newsId=36350,printer.html  
www\_iaaf\_org\_news\_newsId=20600,printer.html  
www\_iaaf\_org\_news\_newsId=22233,printer.html  
www\_iaaf\_org\_news\_newsId=23298,printer.html  
www\_iaaf\_org\_news\_newsId=24613,printer.html  
www\_iaaf\_org\_news\_newsId=27225,printer.html  
www\_iaaf\_org\_news\_newsId=29594,printer.html  
www\_iaaf\_org\_news\_newsId=30271,printer.html  
www\_iaaf\_org\_news\_newsId=30706,printer.html  
www\_iaaf\_org\_news\_newsId=31109,printer.html  
www\_iaaf\_org\_news\_newsId=35211,printer.html  
www\_iaaf\_org\_news\_newsId=35364,printer.html  
www\_iaaf\_org\_news\_newsId=35525,printer.html  
www\_iaaf\_org\_news\_newsId=36080,printer.html  
www\_iaaf\_org\_news\_newsId=36408,printer.html  
www\_iaaf\_org\_news\_newsId=37782,printer.html  
www\_iaaf\_org\_news\_newsId=37969,printer.html  
www\_iaaf\_org\_news\_newsId=38356,printer.html  
www\_iaaf\_org\_news\_newsId=38772,printer.html  
www\_iaaf\_org\_news\_newsId=39116,printer.html  
www\_iaaf\_org\_news\_newsId=39141,printer.html  
www\_iaaf\_org\_news\_newsId=39370,printer.html  
www\_iaaf\_org\_news\_newsId=40160,printer.html

## A.2 Log of the relation extraction example

.. Bungei .... wins ?  
Bungei ---- personNameToRanking ---- wins  
.. Bungei .... men ?  
Bungei ---- personNameToGender ---- men  
.. Bungei .... 800m ?  
multiple relations allowed for PersonName, SportsName  
800m ---- sportsNameToPersonName ---- Bungei  
.. wins .... men ?  
genderToRanking is an invalid relation  
.. wins .... 800m ?  
800m ---- sportsNameToRanking ---- wins  
.. men .... 800m ?  
sportsNameToGender is an invalid relation  
.. London .... July 26th ?  
dateToCity is an invalid relation  
.. London .... Kenya ?  
countryToCity is an invalid relation  
.. London .... 2004 Olympics ?  
2004 Olympics ---- sportsEventNameToCity ---- London  
.. London .... silver ?  
rankingToCity is an invalid relation  
.. London .... Wilfred Bungei ?  
personNameToCity is an invalid relation  
.. London .... 24 ?  
ageToCity is an invalid relation  
.. London .... 1:43.72 ?



performanceToCity is an invalid relation  
.. London .... Denmark ?  
countryToCity is an invalid relation  
.. London .... Wilson Kipketer ?  
personNameToCity is an invalid relation  
.. London .... 1:43.88 ?  
performanceToCity is an invalid relation  
.. London .... semi-finals ?  
sportsRoundNameToCity is an invalid relation  
.. London .... Wembley ?  
stadiumToCity is an invalid relation  
.. July 26th .... Kenya ?  
countryToDate is an invalid relation  
.. July 26th .... 2004 Olympics ?  
2004 Olympics ---- sportsEventNameToDate ---- July 26th  
.. July 26th .... silver ?  
rankingToDate is an invalid relation  
.. July 26th .... Wilfred Bungei ?  
personNameToDate is an invalid relation  
.. July 26th .... 24 ?  
ageToDate is an invalid relation  
.. July 26th .... 1:43.72 ?  
performanceToDate is an invalid relation  
.. July 26th .... Denmark ?  
countryToDate is an invalid relation  
.. July 26th .... Wilson Kipketer ?  
personNameToDate is an invalid relation  
.. July 26th .... 1:43.88 ?  
performanceToDate is an invalid relation  
.. July 26th .... semi-finals ?  
semi-finals ---- sportsRoundNameToDate ---- July 26th  
.. July 26th .... Wembley ?  
stadiumToDate is an invalid relation  
.. Kenya .... 2004 Olympics ?  
2004 Olympics ---- sportsEventNameToCountry ---- Kenya  
.. Kenya .... silver ?  
rankingToCountry is an invalid relation  
.. Kenya .... Wilfred Bungei ?  
Wilfred Bungei ---- personNameToCountry ---- Kenya  
.. Kenya .... 24 ?  
ageToCountry is an invalid relation  
.. Kenya .... 1:43.72 ?  
performanceToCountry is an invalid relation  
.. 2004 Olympics .... silver ?  
rankingToSportsEventName is an invalid relation  
multiple relations allowed for SportsEventName, PersonName  
.. 2004 Olympics .... Wilfred Bungei ?  
2004 Olympics ---- sportsEventNameToPersonName ---- Wilfred Bungei  
.. 2004 Olympics .... 24 ?  
ageToSportsEventName is an invalid relation  
.. 2004 Olympics .... 1:43.72 ?  
performanceToSportsEventName is an invalid relation  
multiple relations allowed for SportsEventName, PersonName  
.. 2004 Olympics .... Wilson Kipketer ?  
2004 Olympics ---- sportsEventNameToPersonName ---- Wilson Kipketer

.. 2004 Olympics .... 1:43.88 ?  
performanceToSportsEventName is an invalid relation  
.. 2004 Olympics .... semi-finals ?  
sportsRoundNameToSportsEventName is an invalid relation  
.. 2004 Olympics .... Wembley ?  
stadiumToSportsEventName is an invalid relation  
.. silver .... Wilfred Bungei ?  
Wilfred Bungei ---- personNameToRanking ---- silver  
.. silver .... 24 ?  
ageToRanking is an invalid relation  
.. silver .... 1:43.72 ?  
1:43.72 ---- performanceToRanking ---- silver  
.. silver .... Denmark ?  
countryToRanking is an invalid relation  
.. silver .... semi-finals ?  
semi-finals ---- sportsRoundNameToRanking ---- silver  
.. silver .... Wembley ?  
stadiumToRanking is an invalid relation  
.. Wilfred Bungei .... 24 ?  
Wilfred Bungei ---- personNameToAge ---- 24  
.. Wilfred Bungei .... 1:43.72 ?  
Wilfred Bungei ---- personNameToPerformance ---- 1:43.72  
multiple relations allowed for PersonName, SportsRoundName  
.. Wilfred Bungei .... semi-finals ?  
semi-finals ---- sportsRoundNameToPersonName ---- Wilfred Bungei  
.. 24 .... 1:43.72 ?  
performanceToAge is an invalid relation  
.. 24 .... Denmark ?  
countryToAge is an invalid relation  
.. 24 .... 1:43.88 ?  
performanceToAge is an invalid relation  
.. 24 .... semi-finals ?  
sportsRoundNameToAge is an invalid relation  
.. 24 .... Wembley ?  
stadiumToAge is an invalid relation  
.. 1:43.72 .... Denmark ?  
countryToPerformance is an invalid relation  
.. Denmark .... Wilson Kipketer ?  
Wilson Kipketer ---- personNameToCountry ---- Denmark  
.. Denmark .... 1:43.88 ?  
performanceToCountry is an invalid relation  
.. Denmark .... semi-finals ?  
sportsRoundNameToCountry is an invalid relation  
.. Denmark .... Wembley ?  
stadiumToCountry is an invalid relation  
.. Wilson Kipketer .... 1:43.88 ?  
Wilson Kipketer ---- personNameToPerformance ---- 1:43.88  
multiple relations allowed for PersonName, SportsRoundName  
.. Wilson Kipketer .... semi-finals ?  
semi-finals ---- sportsRoundNameToPersonName ---- Wilson Kipketer  
.. Wilson Kipketer .... Wembley ?  
stadiumToPersonName is an invalid relation  
.. 1:43.88 .... semi-finals ?  
semi-finals ---- sportsRoundNameToPerformance ---- 1:43.88  
.. 1:43.88 .... Wembley ?

stadiumToPerformance is an invalid relation  
 .. semi-finals .... Wembley ?  
 stadiumToSportsRoundName is an invalid relation

### A.3 Evaluation Set results

newsId=22172,printer.html\_00016.xml

Word count: 1284

Annotation Type	Precision	Recall
PersonName	0.9787234042553192	0.8518518518518519
Gender	0.75	1.0
Age	1.0	1.0
Stadium	1.0	0.0
Performance	0.9090909090909091	0.967741935483871
Country	0.92	0.8846153846153846
City	1.0	0.8461538461538461
Date	1.0	1.0
SportsName	0.6666666666666666	1.0
Ranking	0.7391304347826086	0.7727272727272727
SportsRoundName	0.5	1.0
SportsEventName	0.71875	0.8846153846153846

newsId=22233,printer.html\_00017.xml

Word count: 1469

Annotation Type	Precision	Recall
PersonName	0.9926470588235294	0.9642857142857143
Gender	0.9444444444444444	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9404761904761905	0.9634146341463414
Country	0.967741935483871	0.967741935483871
City	1.0	1.0
Date	1.0	1.0
SportsName	0.7142857142857143	0.9375
Ranking	0.7037037037037037	0.7307692307692307
SportsRoundName	1.0	0.3333333333333333

SportsEventName	0.6538461538461539	0.9444444444444444
-----------------	--------------------	--------------------

### newsId=27380,printer.html\_00018.xml

Word count: 1195

Annotation Type	Precision	Recall
PersonName	1.0	0.9625
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9545454545454546	1.0
Country	0.9153846153846155	0.9916666666666667
City	0.8947368421052632	0.8947368421052632
Date	1.0	1.0
SportsName	0.7236842105263157	0.9821428571428572
Ranking	1.0	0.6701030927835051
SportsRoundName	0.0	0.0
SportsEventName	0.7647058823529411	0.7647058823529411

### newsId=31278,printer.html\_00019.xml

Word count: 710

Annotation Type	Precision	Recall
PersonName	1.0	1.0
Gender	1.0	1.0
Age	0.6666666666666666	1.0
Stadium	1.0	1.0
Performance	0.5	1.0
Country	1.0	1.0
City	1.0	0.8
Date	1.0	0.5
SportsName	1.0	1.0
Ranking	0.8571428571428571	0.5454545454545454
SportsRoundName	0.3333333333333333	0.25
SportsEventName	0.125	0.5

### newsId=31945,printer.html\_0001A.xml

Word count: 1870

Annotation Type	Precision	Recall
PersonName	0.9915966386554622	0.959349593495935
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	0.8888888888888888
Country	0.9603174603174602	0.9453125
City	1.0	0.8666666666666667
Date	1.0	0.6111111111111112
SportsName	0.7317073170731707	1.0
Ranking	0.7674418604651163	0.673469387755102
SportsRoundName	0.5	0.75
SportsEventName	0.41000000000000003	0.7884615384615384

### newsId=33708,printer.html\_0001B.xml

Word count: 1758

Annotation Type	Precision	Recall
PersonName	0.975609756097561	0.8602150537634409
Gender	0.875	1.0
Age	1.0	1.0
Stadium	1.0	0.0
Performance	0.9871794871794872	0.9058823529411765
Country	0.9402985074626866	0.8873239436619719
City	1.0	0.6590909090909091
Date	1.0	0.875
SportsName	0.8	1.0
Ranking	0.8823529411764706	0.4838709677419355
SportsRoundName	1.0	1.0
SportsEventName	0.775	0.8611111111111112

### newsId=35525,printer.html\_0001C.xml

Word count: 1046

Annotation Type	Precision	Recall
PersonName	0.9953271028037383	0.9681818181818183
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0

Performance	1.0	1.0
Country	0.9772727272727273	1.0
City	1.0	1.0
Date	1.0	1.0
SportsName	0.6779661016949152	1.0
Ranking	0.9841269841269841	0.8211920529801324
SportsRoundName	1.0	1.0
SportsEventName	1.0	1.0

### newsId=35884,printer.html\_0001D.xml

Word count: 1510

Annotation Type	Precision	Recall
PersonName	0.9891304347826086	0.9680851063829787
Gender	0.8888888888888888	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9354838709677419	0.9666666666666667
Country	0.85	0.918918918918919
City	1.0	0.9444444444444444
Date	1.0	0.5
SportsName	0.5454545454545454	0.9230769230769231
Ranking	0.875	0.6222222222222222
SportsRoundName	1.0	1.0
SportsEventName	0.8333333333333334	0.9615384615384616

### newsId=36080,printer.html\_0001E.xml

Word count: 920

Annotation Type	Precision	Recall
PersonName	0.9883720930232558	0.8673469387755102
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	1.0
Country	0.95625	0.9935064935064934
City	1.0	1.0
Date	0.75	0.75
SportsName	0.6190476190476191	0.9285714285714286

Ranking	0.9918032786885246	0.8461538461538461
SportsRoundName	1.0	1.0
SportsEventName	0.7	0.875

### newsId=36350,printer.html\_0001F.xml

Word count: 956

Annotation Type	Precision	Recall
PersonName	0.9789473684210526	0.9299999999999999
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	1.0
Country	0.968421052631579	1.0
City	1.0	0.8333333333333334
Date	1.0	1.0
SportsName	0.75	0.9069767441860465
Ranking	0.984251968503937	0.9057971014492754
SportsRoundName	0.0	1.0
SportsEventName	0.875	0.875

### www\_iaaf\_org\_news\_newsId=20600,printer.html\_00020.xml

Word count: 2197

Annotation Type	Precision	Recall
PersonName	0.98	0.9074074074074074
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	0.8888888888888888
Country	0.8421052631578947	0.8421052631578947
City	1.0	0.8
Date	1.0	1.0
SportsName	0.78	0.975
Ranking	0.7272727272727273	0.8888888888888888
SportsRoundName	1.0	0.0
SportsEventName	0.9761904761904762	0.9761904761904762

**www\_iaaf\_org\_news\_newsId=22233,printer.html\_00021.xml**

Word count: 1468

<b>Annotation Type</b>	<b>Precision</b>	<b>Recall</b>
PersonName	0.9926470588235294	0.9642857142857143
Gender	0.9444444444444444	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9404761904761905	0.9634146341463414
Country	0.967741935483871	0.967741935483871
City	1.0	1.0
Date	1.0	1.0
SportsName	0.7142857142857143	0.9375
Ranking	0.7857142857142857	0.7857142857142857
SportsRoundName	1.0	0.3333333333333333
SportsEventName	0.6538461538461539	0.9444444444444444

**www\_iaaf\_org\_news\_newsId=23298,printer.html\_00022.xml**

Word count: 1058

<b>Annotation Type</b>	<b>Precision</b>	<b>Recall</b>
PersonName	0.8819444444444444	0.9338235294117647
Gender	1.0	1.0
Age	0.5	0.5
Stadium	1.0	1.0
Performance	0.9871794871794872	0.9166666666666667
Country	1.0	0.9642857142857143
City	0.7142857142857143	0.625
Date	1.0	1.0
SportsName	0.0	1.0
Ranking	1.0	0.9696969696969697
SportsRoundName	1.0	1.0
SportsEventName	0.9	0.9

**www\_iaaf\_org\_news\_newsId=24613,printer.html\_00023.xml**

Word count: 1165

<b>Annotation Type</b>	<b>Precision</b>	<b>Recall</b>
PersonName	0.8656716417910448	0.8656716417910448



Gender	0.6666666666666666	1.0
Age	0.5	1.0
Stadium	1.0	0.0
Performance	0.7666666666666666	0.8846153846153846
Country	0.8409090909090909	0.9487179487179487
City	0.59375	0.6785714285714286
Date	0.8214285714285714	0.8846153846153846
SportsName	0.2222222222222222	0.6666666666666666
Ranking	0.8148148148148148	0.9565217391304348
SportsRoundName	1.0	1.0
SportsEventName	0.8125	0.8125

### www\_iaaf\_org\_news\_newsId=27225,printer.html\_00024.xml

Word count: 1224

Annotation Type	Precision	Recall
PersonName	0.9318181818181819	0.9461538461538461
Gender	1.0	1.0
Age	0.75	1.0
Stadium	1.0	0.0
Performance	0.8666666666666667	1.0
Country	0.9666666666666667	0.9354838709677419
City	1.0	0.5714285714285714
Date	1.0	0.6
SportsName	0.5333333333333333	0.8
Ranking	0.8260869565217391	0.475
SportsRoundName	0.6666666666666666	1.0
SportsEventName	0.5833333333333334	0.6363636363636364

### www\_iaaf\_org\_news\_newsId=29594,printer.html\_00025.xml

Word count: 1309

Annotation Type	Precision	Recall
PersonName	0.9746835443037976	0.8651685393258427
Gender	1.0	1.0
Age	1.0	1.0
Stadium	0.25	0.5
Performance	0.9824561403508771	1.0
Country	0.9272727272727272	0.8793103448275862

City	1.0	0.8918918918918919
Date	0.7619047619047619	0.7619047619047619
SportsName	0.7	1.0
Ranking	0.875	0.6774193548387096
SportsRoundName	0.6	1.0
SportsEventName	0.6826923076923077	0.9342105263157895

**www\_iaaf\_org\_news\_newsId=30271,printer.html\_00026.xml**

Word count: 795

Annotation Type	Precision	Recall
PersonName	1.0	0.9019607843137255
Gender	1.0	1.0
Age	0.7142857142857143	1.0
Stadium	1.0	1.0
Performance	0.9375	0.9090909090909092
Country	0.8461538461538461	0.9166666666666666
City	0.8	0.8
Date	0.6666666666666666	0.6666666666666666
SportsName	0.8571428571428571	1.0
Ranking	1.0	0.9090909090909091
SportsRoundName	1.0	1.0
SportsEventName	0.2	1.0

**www\_iaaf\_org\_news\_newsId=30706,printer.html\_00027.xml**

Word count: 2072

Annotation Type	Precision	Recall
PersonName	0.9227272727272727	0.9227272727272727
Gender	0.8333333333333334	1.0
Age	1.0	0.4
Stadium	1.0	1.0
Performance	0.9921875	0.976923076923077
Country	0.9726027397260274	0.9726027397260274
City	0.875	0.9130434782608695
Date	1.0	0.8
SportsName	0.6333333333333333	0.890625
Ranking	0.92	0.7796610169491526
SportsRoundName	1.0	1.0

SportsEventName	0.6527777777777778	0.8703703703703703
-----------------	--------------------	--------------------

**www\_iaaf\_org\_news\_newsId=31109,printer.html\_00028.xml**

Word count: 697

Annotation Type	Precision	Recall
PersonName	1.0	0.8695652173913043
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	1.0
Country	1.0	0.5454545454545454
City	1.0	0.6666666666666666
Date	0.5	0.5
SportsName	1.0	1.0
Ranking	0.7777777777777778	0.6363636363636364
SportsRoundName	0.5	0.5
SportsEventName	0.5555555555555556	0.7142857142857143

**www\_iaaf\_org\_news\_newsId=35211,printer.html\_00029.xml**

Word count: 822

Annotation Type	Precision	Recall
PersonName	0.9761904761904762	0.8913043478260869
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9245283018867925	0.98
Country	0.9259259259259259	0.9615384615384616
City	0.8181818181818182	0.8181818181818182
Date	1.0	1.0
SportsName	0.6666666666666666	1.0
Ranking	0.9545454545454546	0.84
SportsRoundName	0.6666666666666666	1.0
SportsEventName	0.5714285714285714	1.0

**www\_iaaf\_org\_news\_newsId=35364,printer.html\_0002A.xml**

Word count: 1642

Annotation Type	Precision	Recall
PersonName	0.975	0.9466019417475728
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9692307692307692	0.9692307692307692
Country	0.9473684210526315	0.9642857142857143
City	0.92	0.9583333333333334
Date	0.9	0.75
SportsName	0.8571428571428571	0.967741935483871
Ranking	0.896551724137931	0.6842105263157895
SportsRoundName	0.0	1.0
SportsEventName	0.5555555555555556	0.9375

### www\_iaaf\_org\_news\_newsid=35525,printer.html\_0002B.xml

Word count: 1046

Annotation Type	Precision	Recall
PersonName	0.9953271028037383	0.9681818181818183
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	1.0
Country	0.9772727272727273	1.0
City	1.0	1.0
Date	1.0	1.0
SportsName	0.6779661016949152	1.0
Ranking	1.0	0.8235294117647058
SportsRoundName	1.0	1.0
SportsEventName	1.0	1.0

### www\_iaaf\_org\_news\_newsid=36080,printer.html\_0002C.xml

Word count: 920

Annotation Type	Precision	Recall
PersonName	0.9883720930232558	0.8673469387755102
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0

Performance	1.0	1.0
Country	0.95625	0.9935064935064934
City	1.0	1.0
Date	1.0	1.0
SportsName	0.6190476190476191	0.9285714285714286
Ranking	1.0	0.8472222222222222
SportsRoundName	1.0	0.0
SportsEventName	0.6000000000000001	0.75

### www\_iaaf\_org\_news\_newsId=36408,printer.html\_0002D.xml

Word count: 498

Annotation Type	Precision	Recall
PersonName	0.8888888888888888	0.9142857142857144
Gender	1.0	1.0
Age	0.5	0.5
Stadium	1.0	1.0
Performance	1.0	1.0
Country	1.0	1.0
City	1.0	1.0
Date	1.0	1.0
SportsName	1.0	1.0
Ranking	0.967741935483871	0.7142857142857143
SportsRoundName	1.0	1.0
SportsEventName	0.9	0.9

### www\_iaaf\_org\_news\_newsId=37782,printer.html\_0002E.xml

Word count: 1339

Annotation Type	Precision	Recall
PersonName	0.9127906976744187	0.9127906976744187
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9245283018867925	0.98
Country	0.8064516129032258	0.8333333333333334
City	0.8571428571428571	0.6923076923076923
Date	0.9166666666666667	0.55
SportsName	0.6363636363636364	1.0

Ranking	0.8974358974358975	0.9210526315789473
SportsRoundName	0.0	0.0
SportsEventName	0.5142857142857142	0.8571428571428572

### www\_iaaf\_org\_news\_newsId=37969,printer.html\_0002F.xml

Word count: 669

Annotation Type	Precision	Recall
PersonName	0.8367346938775511	0.8913043478260869
Gender	1.0	1.0
Age	0.5	0.5
Stadium	1.0	1.0
Performance	0.96	1.0
Country	0.8571428571428571	0.8888888888888888
City	0.9615384615384616	0.9615384615384616
Date	1.0	0.56
SportsName	0.5	1.0
Ranking	1.0	0.6896551724137931
SportsRoundName	1.0	1.0
SportsEventName	0.467741935483871	0.90625

### www\_iaaf\_org\_news\_newsId=38356,printer.html\_00030.xml

Word count: 949

Annotation Type	Precision	Recall
PersonName	0.9324324324324325	0.7840909090909091
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	1.0	1.0
Country	1.0	1.0
City	0.9375	0.9375
Date	1.0	1.0
SportsName	0.5	0.5
Ranking	0.875	0.875
SportsRoundName	1.0	1.0
SportsEventName	0.75	0.9545454545454546

**www\_iaaf\_org\_news\_newsId=38772,printer.html\_00031.xml**

Word count: 1122

<b>Annotation Type</b>	<b>Precision</b>	<b>Recall</b>
PersonName	0.972972972972973	0.96
Gender	1.0	1.0
Age	1.0	0.8
Stadium	1.0	1.0
Performance	0.8795180722891566	0.9864864864864865
Country	0.7307692307692307	0.9743589743589743
City	0.8461538461538461	0.5
Date	1.0	1.0
SportsName	0.6785714285714286	0.95
Ranking	0.9032258064516129	0.8484848484848485
SportsRoundName	0.0	1.0
SportsEventName	0.6538461538461539	0.85

**www\_iaaf\_org\_news\_newsId=39116,printer.html\_00032.xml**

Word count: 1885

<b>Annotation Type</b>	<b>Precision</b>	<b>Recall</b>
PersonName	0.9946808510638299	0.9444444444444444
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	1.0
Performance	0.9519230769230769	0.9705882352941176
Country	0.8947368421052632	0.9622641509433962
City	0.8666666666666667	0.8666666666666667
Date	1.0	0.8333333333333334
SportsName	0.5487804878048781	0.9375
Ranking	0.7567567567567568	0.9032258064516129
SportsRoundName	0.0	0.0
SportsEventName	0.3064516129032258	0.8636363636363636

**www\_iaaf\_org\_news\_newsId=39141,printer.html\_00033.xml**

Word count: 984

<b>Annotation Type</b>	<b>Precision</b>	<b>Recall</b>
PersonName	0.9130434782608696	0.9767441860465116

Gender	0.8666666666666667	1.0
Age	0.875	1.0
Stadium	1.0	0.0
Performance	0.9861111111111112	0.9861111111111112
Country	0.8571428571428571	1.0
City	1.0	0.7
Date	0.9285714285714286	0.9285714285714286
SportsName	0.5263157894736842	1.0
Ranking	0.9629629629629629	0.9629629629629629
SportsRoundName	0.4	0.5
SportsEventName	0.6000000000000001	0.8823529411764706

### www\_iaaf\_org\_news\_newsId=39370,printer.html\_00034.xml

Word count: 1524

Annotation Type	Precision	Recall
PersonName	0.990990990990991	0.9821428571428572
Gender	1.0	1.0
Age	1.0	1.0
Stadium	1.0	0.0
Performance	0.948051948051948	0.9358974358974359
Country	0.8333333333333334	0.9302325581395349
City	1.0	0.3636363636363636
Date	1.0	0.5
SportsName	0.5714285714285714	0.9230769230769231
Ranking	0.9285714285714286	0.7027027027027027
SportsRoundName	0.2	0.5
SportsEventName	0.3157894736842105	1.0

### www\_iaaf\_org\_news\_newsId=40160,printer.html\_00035.xml

Word count: 1051

Annotation Type	Precision	Recall
PersonName	0.8870967741935484	0.5729166666666667
Gender	1.0	1.0
Age	1.0	0.8333333333333334
Stadium	1.0	1.0
Performance	0.8421052631578947	1.0
Country	0.6666666666666666	1.0



City	0.9	0.8181818181818182
Date	0.9	1.0
SportsName	0.6875	1.0
Ranking	0.8333333333333334	0.9090909090909091
SportsRoundName	0.625	0.5
SportsEventName	0.6666666666666667	0.8888888888888888

## A.4 JAPE Rules for named entity “Performance”

**Phase:** Performance //mustn't overlap with "Date"

**Input:** Token Lookup PersonName Split //must be in one sentence.

**Options:** control = appelt

**Macro:** AND

```
(
  {Token.root=="and"} |
  {Token.root==" , "} //used for terms like "two hours, three minutes and 1 second"
)
```

**Macro:** PERF //length

```
(
  (ONE_DIGIT|TWO_DIGIT)
  ({Token.root=="."})
  (ONE_DIGIT|TWO_DIGIT) // 1.3, 24.5
)
```

**Macro:** TIME\_PERF //time

```
(
  (ONE_DIGIT|TWO_DIGIT)
  ({Token.root==":"})
  (ONE_DIGIT|TWO_DIGIT) // 3:45
)
```

**Macro:** NUM\_OR\_PERF

```
(
  NUMBER_WORDS | PERF //twenty-two; 2.3
)
```

**Macro:** TIME\_UNIT

```
(
  {Token.root=="s"} |
  {Token.root=="sec"} |
  {Token.root=="second"} |
  {Token.root=="seconds"} |
  {Token.root=="min"} |
  {Token.root=="minute"} |
  {Token.root=="h"} |
  {Token.root=="hr"} |
  {Token.root=="hour"}
)
```

**Macro:** LENGTH\_UNIT

```
(
```

```

{Token.root=="m"}|
{Token.root=="meter"}|
{Token.root=="metre"}|
{Token.root=="foot"}|
{Token.root=="feet"}|
{Token.root=="ft"}
)

////////////////////////////////////

Rule: notSportsName
Priority: 1000
({Lookup.majorType==SportsName})
-->
{} //do nothing here, just prevent other rules from matching

Rule: notTime
Priority: 1000
({Token.root=="at"})?
(TIME_PERF (TIME_AMPM | TIME_ZONE) ) // 8:30 am; 8:30(GMT)
-->
{} //do nothing here, just prevent other rules from matching

Rule: notRoundName // Heat 1: 1. Smith
Priority: 1000
(
  ({Token.root=="race"}|{Token.root=="heat"} )
  (TIME_PERF)
)
-->
{}

//////////////////////////////////// WIND - negative rules //////////////////////////////////////

Rule: NegativeRule2 //wind
Priority: 200
(
  {Token.root=="w"} //w:0.7
  {Token.root==":"}
  ({Token.root=="-"})?
  PERF
)
-->
{} //do nothing

Rule: NegativeRule3 // "1.4 m/s"
Priority: 200
(
  ({Token.root=="-"})?
  PERF
  {Token.root=="m"}{Token.root=="/"}{Token.root=="s"} //must have
)
-->
{} //do nothing

Rule: NegativeRule4 // "-1.4"
(
  {Token.root=="-"} //must have
  PERF
  ({Token.root=="m"}{Token.root=="/"} {Token.root=="s"})?
)
-->
{} //do nothing

Rule: NegativeRule5 // headwind of 1.3; tailwind of 0.5m/s
Priority: 200

```

```
(
  (
    {Token.root=="headwind"}|
    {Token.root=="tailwind"}|
    {Token.root=="wind"}
  )
  ({Token.root=="of"})?
  PERF
  ({Token.root=="m"}{Token.root=="/"} {Token.root=="s"})?
)
-->
{} //do nothing
```

**Rule:** NegativeRule6 // 1.3 wind; 0.5m/s of tailwind  
**Priority:** 200

```
(
  PERF
  ({Token.root=="m"}{Token.root=="/"} {Token.root=="s"})?
  ({Token.root=="of"})?
  (
    {Token.root=="headwind"}|
    {Token.root=="tailwind"}|
    {Token.root=="wind"}
  )
)
-->
{} //do nothing
```

```
////////////////////////////////////
////////////////////////////////////
```

**Rule:** NegativeRule7 // over 2.00; under 3 seconds  
**Priority:** 200

```
(
  (
    {Token.root=="over"}|
    {Token.root=="under"}|
    {Token.root=="sub"}|
    {Token.root=="almost"}
  )
  (PERF | NUMBER_WORDS | TIME_PERF) //two meters , 3:45
  (LENGTH_UNIT|TIME_UNIT)? //m, meter, second..
)
-->
{} //do nothing
```

**Rule:** NegativeRule8 // 0.23 outside his season's best  
**Priority:** 200

```
(
  (
    (
      NUM_OR_PERF
      (LENGTH_UNIT)?
    ) //0.23; 1.23meters ; 2 feet two meters
    |
    (
      NUM_OR_PERF TIME_UNIT //twenty-two seconds clear ; 1.2 seconds ahead
      (AND NUM_OR_PERF TIME_UNIT)? //2 hours and three minutes
      (AND NUM_OR_PERF TIME_UNIT)? //2 hours, three minutes and 5 seconds
    )
  )
  (
    {Token.root=="over"}|
    {Token.root=="under"}|
    {Token.root=="outside"}| //2.00 clear off his rivals
    {Token.root=="clear"}|
  )
)
```

```

    {Token.root=="ahead"}|
    {Token.root=="before"}|
    {Token.root=="behind"}
  )
)
-->
{} //do nothing here, just prevent other rules from matching

Rule: NotPerfMeters //nobody can jump or throw a hammer further than 100meters
Priority: 100
(
  {Token.kind==number, Token.length>2} //100m
  ({Token.root == "m"}|{Token.root == "meter"}|{Token.root == "metre"}) //meters
)
-->
{} //do nothing here, just prevent other rules from matching

////////////////////////////////////
////////////////////////////////////

Rule: PerfTime1
Priority: 30
// 1:29:00.6
// 1:43.76
(
  ({Token.root == "("})? //optional opening bracket
  (
    (ONE_DIGIT|TWO_DIGIT)
    {Token.string == ":"}
    (ONE_DIGIT|TWO_DIGIT) //1:29
    (
      {Token.string == ":"} //1:29:29
      (ONE_DIGIT|TWO_DIGIT)
    )?
    (
      {Token.string == "."} // 1:29.06
      (ONE_DIGIT|TWO_DIGIT)
    )?
  ):time
  ({Token.root == ")"})? //optional closing bracket
)
-->
:time.Performance = {class="Performance", rule = "PerfTime1"}

////////////////////////////////////

Rule: PerfMeters
(
  (PERF):meters
  (LENGTH_UNIT) //meters
)
-->
:meters.Performance = {class = "Performance", rule = "PerfMeters"}

Rule: PerfMetersCont
(
  (
    {Token.root=="in"}| //in 2.34
    {Token.root=="with"}
  )
  (PERF):meters
  (LENGTH_UNIT)? //optional
  ({Token.root=="over"})? //won in 2.34 over the competition
)

```

```

)
-->
:meters.Performance = {class = "Performance", rule = "PerfMetersCont"}

Rule: PerfInches
/**** /18-8.75 */
/**** /19-2 */
/**** /19-00.25 */

(
  {Token.root == "/" }
  (
    (ONE_DIGIT|TWO_DIGIT) //always an integer
    {Token.root == "-"} //always
    (ONE_DIGIT|TWO_DIGIT) //always
    (
      {Token.root == "."} //optional
      (ONE_DIGIT|TWO_DIGIT)
    )?
  ):inches
  ({Token.root == "}")? //optional closing bracket
)
-->
:inches.Performance = {class = "Performance", rule = "PerfInches"}

Rule: PerfVerb
(
  (
    {Token.root == "clear"}|
    {Token.root == "clock"}
  )
  (PERF):perf
  (LENGTH_UNIT | TIME_UNIT)? // clear 4.80m
)
-->
:perf.Performance = {class = "Performance", rule = "PerfVerb"}

////////////////////////////////////////////////////////////////

Rule: PerfTimeContext
Priority: 30
(
  {Token.string == "in"}
)?
(
  (ONE_DIGIT|TWO_DIGIT)
  ({Token.string == ":"}|{Token.string == "."}) // in 2:23 minutes
  (TWO_DIGIT)
  (TIME_UNIT)
):time
-->
:time.Performance = {class = "Performance", rule = "PerfTimeContext"}

// Performance rules, e.g. 2.03m, 35s
// time performance is unambiguous, e.g. 1:45.25
Rule: PerformanceNumbers
Priority: 30
(
  ({Token.string == "at"})? //<he made a try "at 4.60m">.
  (
    (
      {Token.kind == number}
      {Token.string == ":"}
    )?
  )
)

```

```

    {Token.kind == number} //just a number, like 200
  (
    {Token.string == ".", !Split} // commas are used only as separators
    {Token.kind == number}
  )
):numbers
(LENGTH_UNIT | TIME_UNIT)? // don't include units in the annotation pattern
)
-->
:numbers.Performance = { class = "Performance", rule = "PerformanceNumbers" }

Rule: PerformanceWords //twenty-five (meters|seconds)
Priority: 30
(
  ({Token.string == "at"})? //<<he made a try "at 4.60m">>.
  (
    ((NUMBER_WORDS))//if numberwords are used, then the units are included in
annotation
    ( LENGTH_UNIT | TIME_UNIT)
  ):numbers //two meters instead of just "two"
)
-->
:numbers.Performance = { class = "Performance", rule = "PerformanceWords" }

////////////////////////////////////

Rule: PersonToPerf // Feofanova - 4.80
Priority: 220
(
  {Lookup.majorType=="PersonName"} // Feofanova - 4.80 should be recognized
  {Token.root=="-"} //must have
  (PERF):cur
  ({Token.root=="m"})?
)
-->
:cur.Performance={ class = "Performance", rule = "PersonToPerf" }

////////////////////////////////////

Rule: PerfAbbreviation // VIC, QLD, NSW..
Priority: 220 //NSW 3214
( //ACT, NSW, QLD, SA, TAS, VIC, WA, NT.
  (
    {Token.string=="VIC"}|
    {Token.string=="QLD"}|
    {Token.string=="TAS"}|
    {Token.string=="NSW"}|
    {Token.string=="ACT"}|
    {Token.string=="SA"}|
    {Token.string=="NT"}|
    {Token.string=="WA"}
  )
  ({Token.kind==number, Token.length>2}):cur
)
-->
:cur.Performance={ class = "Performance", rule = "PerfAbbreviation" }

```

# Bibliography

- [1] K. Biatov, P. Fragkou, V. Karkaletsis, Alfio Ferrara, and Atila Kaya. Boemie deliverable 2.3 - semantics extraction from non-visual content tools: state-of-the-art report, March 2007.
- [2] Kalina Bontcheva, Danica Damljanovic, and Niraj Aswani. Tao deliverable 3.1 - key concept identification and clustering of similar content. 2004.
- [3] J.A. Borsje. Rule based semi-automatic ontology learning. Master's thesis, Erasmus University Rotterdam, July 2007.
- [4] Valentina Cordi, Paolo Lombardi, Maurizio Martelli, and Viviana Mascardi. An ontology-based similarity between sets of concepts, 2004.
- [5] H. Cunningham. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.
- [6] Hamish Cunningham. *Software Architecture for Language Engineering*. PhD thesis, University of Sheffield, 2000. <http://gate.ac.uk/>.
- [7] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: An architecture for development of robust nlp applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics(ACL)*, pages 168–175, 2002.
- [8] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE User Guide. <http://gate.ac.uk/>, 2008.
- [9] K. Dalakleidi, S. Dasiopoulou, E. Giannakidou, and A. Kaya. Boemie deliverable 3.2 - domain ontologies v1, February 2007.
- [10] K. Dalakleidi, C. Evangelou, S. Dasiopoulou, and V. Tzouvaras. Boemie deliverable 3.5 - domain ontologies v2, August 2007.
- [11] M. Dimitrov. A light-weight approach to coreference resolution for named entities in text. Master's thesis, University of Sofia, Bulgaria.
- [12] Pavlina Fragkou, Georgios Petasis, Aris Theodorakos, Vangelis Karkaletsis, and Constantine D. Spyropoulos. Boemie ontology-based text annotation tool.
- [13] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas D. Edwards. Interpretation as abduction. In *26th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 95–103, Buffalo, New York, 1988.
- [14] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 20(1):19–62, May 2005.
- [15] Xing Jiang and Ah-Hwee Tan. Mining ontological knowledge from domain-specific text documents. In *ICDM '05: Proceedings of the Fifth IEEE*

- International Conference on Data Mining*, pages 665–668, Washington, DC, USA, 2005. IEEE Computer Society.
- [16] Tom Kenter and Diana Maynard. Using GATE as an Annotation Tool, January 2005.
- [17] Thomas Klingberg. Ein system zum inhaltlichen erschließen hierarchisch angeordneter textfragmente. Diplomarbeit, TU Hamburg-Harburg, January 2007.
- [18] Tine Lassen and Thomas Vestskov Terney. An ontology-based approach to disambiguation of semantic relations.
- [19] Y. Li, K. Bontcheva, and H. Cunningham. Hierarchical, perceptron-like learning for ontology based information extraction. In *16th International World Wide Web Conference (WWW2007)*, pages 777–786, 2007.
- [20] Alexander Maedche and Steffen Staab. The text-to-onto ontology learning environment. In *Software Demonstration at ICCS-2000 - Eight International Conference on Conceptual Structures*, 2000.
- [21] Diana Maynard. Metrics for evaluation of ontology-based information extraction. In *In WWW 2006 Workshop on "Evaluation of Ontologies for the Web*, 2006.
- [22] Diana Maynard, Valentin Tablan, Cristian Ursu, Hamish Cunningham, and Yorick Wilks. Named entity recognition from diverse text types. In *In Recent Advances in Natural Language Processing 2001 Conference, Tzigov Chark*, 2001.
- [23] R. Möller and B. Neumann. Ontology-based Reasoning Techniques for Multimedia Interpretation and Retrieval. In *Semantic Multimedia and Ontologies : Theory and Applications*. Springer, 2008.
- [24] Daniel Naber. A rule-based style and grammar checker. Master's thesis, University of Bielefeld, August 2003.
- [25] Nadeau, David, Sekine, and Satoshi. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007.
- [26] UC Berkeley's School of Information Management and Systems. How much information. <http://www2.sims.berkeley.edu/research/projects/how-much-info-2003/execsum.htm#summary>.
- [27] Patrick Pantel. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 113–120, 2006.
- [28] Katerina Pastra, Diana Maynard, Oana Hamza, Hamish Cunningham, and Yorick Wilks. How feasible is the reuse of grammars for named entity recognition. In *In Proceedings of the 3rd Conference on Language Resources and Evaluation (LREC), Canary Islands*, 2002.
- [29] S. Espinosa Peraldi, A. Kaya, S. Melzer, and R. Möller. On ontology based abduction for text interpretation. In A. Gelbukh, editor, *Proc. of 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2008)*, number 4919 in LNCS, pages 194–205. Springer, 2008.
- [30] Georgios Petasis, Vangelis Vangelis Karkaletsis, Georgios Paliouras, and Constantine D. Spyropoulos. Learning context-free grammars to extract relations from text. In *BOEMIE Project*, 2008.



- [31] S. Petridis, N. Tsapatsoulis, D. Kosmopoulos, Y. Pratikakis, V. Gatos, S. Perantonis, G. Petasis, P. Fragou, V. Karkaletsis, K. Biatov, C. Seibert, S. Espinosa, S. Melzer, A. Kaya, and R. Moeller. Boemie deliverable d2.1 - methodology for semantics extraction from multimedia content, December 2006.
- [32] Irena Spasic, Goran Nenadic, and Sophia Ananiadou. In *Proceedings of the 2003 Workshop on Natural Language Processing in Biomedicine*, pages 17–24.
- [33] Ting Wang, Yaoyong Li, Kalina Bontcheva, Hamish Cunningham, and Ji Wang. Automatic extraction of hierarchical relations from text. pages 215–229. 2006.
- [34] Chun Xiao, Dietmar Rösner, and Universität Magdeburg. Finding high-frequent synonyms of a domain-specific verb in english sub-language of medline abstracts using wordnet, 2003.
- [35] A Yakushiji, Y Tateisi, Y Miyao, and J Tsujii. Event extraction from biomedical papers using a full parser. *Pac. Symp. Biocomput*, (6):7.