CoolNELLI [1]

The Cool Named Entity Linker for Linguistic Interaction

Tim van Erven Ori Garin Gustaaf Haan Joeri Honnef
Paul Manchego Peter van der Meer Daan Vreeswijk
Janneke van der Zwaan

February 5, 2005

[1]Project supervisors: Karin Müller, Erik Tjong Kim Sang

Contents

Chapter 1

Introduction

1.1 Goal of the project

The task we set ourselves in the Language and Technology Project is to develop a program that can link words in a given text to relevant and informative web pages. The idea is that an Internet user, let's call her Sam, reading an e-mail, a webpage or any other electronic text, might encounter words that she doesn't understand or names that she would want to have more information about. By way of an example:

> President Bush wrestled caimans on the Galapagos Islands. Mboto Afrizal from General Motors had to rescue him.

Reading this text, an Internet user might want to know more about president Bush: does he look like a caiman wrestler? Did he do any caiman wrestling in his career? Besides that, she might want to know what caimans are, and whether they actually occur on the Galapagos Islands. Finally, she most probably has never heard of Mboto Afrizal. For her, it would be very interesting to have the text linked as:

> President Bush wrestled caimans on the Galapagos Islands. Mboto Afrizal from General Motors had to rescue him.

Where the links refer to (respectively):

- http://en.wikipedia.org/wiki/George_W._Bush
- http://en.wiktionary.org/wiki/Caiman
- http://en.wikipedia.org/wiki/Galapagos_Islands
- http://www.MbotoAfrizal.be
- http://www.GeneralMotors.com

This is exactly what our program should do. We will design a web interface where people can input their text-to-be-linked, and even implement it in an e-mail client.

1.2 Problems & Decisions

First of all, we had to design an overall structure of the program that would be technically useful, and at the same time provide a workable division of labour. The structure we agreed on is discussed in the next section.

1.2.1 Communication between Modules

Before building modules separately, it is of course important to know exactly what kind of information modules expect to get from each other. Deciding on this question in an early stage is not trivial, because it forces one to make crucial choices on the very essential level of what the program will do and how it will work. Because we felt that we might want to come back on some of these decisions, we agreed to use an easily extendible XML format as means of communication.

At first we assumed that the context in which a Named Entity (NE) appears, would be indispensable for finding a relevant web page because most NE's would need to be disambiguated. However, we found out very soon that ambiguous NE's are not the biggest problem, and that it is very hard to automatically find contextual information that can help disambiguation in such exceptional cases as George Bush sr/jr, Lance/Louis/Neil Armstrong, etc. Only very clear identification words such as titles could be of help. But in most cases, for an Internet user it is more informative to know what a NE refers to *outside* a specific context.

Still, for information retrieval on NE's it would be very useful to have some more information than the word alone, to be able to refine the search. The only way to get this information was by making use of built-in databases that could recognize common names, and distinguish between types of NE's. How the NE recognition uses these databases, and how the information is used in information retrieval, is discussed in the chapters of the respective modules.

1.2.2 Name

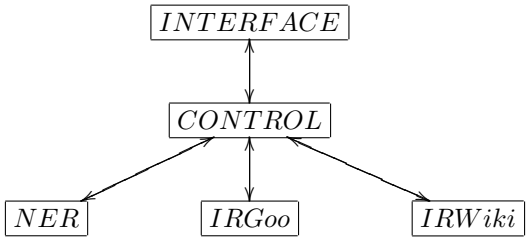
Programs need a name. The easiest way is gluing some acronym together, like Named Entity Linker for Linguistic Information, NELLI. But since this does not sound very flashy, we agreed on the more commercial CoolNELLI.

1.2.3 Structure

The most straightforward option for the program architecture was to divide the operations on the text in three modules connected in serial:

> INPUT ▷ Named Entity Recognition ▷ Information Retrieval ▷ Interface ▷ OUTPUT

Considering the Information Retrieval module however, we decided that it would be better to split its task into two search methods: Google (see Chapter 3) and Wikipedia (see Chapter 4). For the structure this meant that two modules would have an equivalent function, and that we needed some higher level module to control their input and output. Therefore we agreed to build the program according to the following structure:



The advantage of having an independent Control Module would further be that the program could more easily be integrated in and applied to other systems, such as an e-mail client or a web server.

1.3 Evaluation

The only standard to evaluate a system like ours is of course the Average Internet User. Unfortunatly, she was not available for CoolNELLI, and we had to come up with a less absolute measure.

We put a test set together consisting of electronic texts of different genres: a news article on Balkenende visiting the White House, an MTV announcement of some Elvis Tribute featuring Justin Timberlake, a encyclopedia entry on the Galapagos Islands and a business e-mail (see Chapter 7). All project members individually read the texts. We manually marked words in these texts if we thought it would be useful to have more information on it. Above that, for every marked word we sought the most relevant link on the Internet. In general, the results differed only on details, so we could combine them to a standard.

We will use this standard to evaluate our system. In our evaluation we will consider the following parameters:

- Recall: the number of correctly identified Named Entities divided by the number of wanted Named Entities;
- Precision: the number of correctly identified Named Entities divided by the total number of outputted Named Entities;
- The percentage of correct (exact) Wikipedia links;
- The percentage of correct (useful) Google links;
- The processing speed of the system.

.

Chapter 2

Named Entity Recognition

Ori Garin & Gustaaf Haan

2.1 Goals

First of all, the task of our module is of course to recognize words that could be linked to relevant, informative web pages[1]. Above that, we have to bear in mind the goal of the overall program, and enrich our output in such a way that the other modules will be able to give optimal results. The extra requirements per module:

- The *Control Module* needs to know not only what words in the text are linkable, but also where to find them. We have to provide the exact offsets.
- For refining the search, the *IR Google Module* wants to know which *type* of word it is dealing with.
- Where the IR Google is mainly interested in proper names, the *IR Wiki Module* can also explain the meaning of other uncommon words, by referring to the entries of a built-in dictionary. We have to find a way to identify such words.
- Since the *Interface Module* does not want to annoy the user with an overload of links, we have to make sure that different occurrences of one word are linked only once.

2.2 Linkable Word Recognition

Before being able to find Linkable Words, we had to find criteria for linkability. The most straightforward option was to search for Named Entities. In the literature, Named Entities are phrases that contain names of persons, organisations, locations, times and quantities. For the latter two, it is hard to find relevant and informative web pages. Considering CoolNELLI's objective we therefore decided to focus on proper names of persons, locations and organisations, and in a later stage to extend our method to infrequent words.

[1] A better name for our module would have been: Linkable Words Recognition (LWR), since we not only identify NEs.

2.2.1 How to identify Named Entities?

Usually, a named entity will be a group of one or more consecutive capitalized proper names. The following approaches were tried:

POS tagger With this approach, each word is typically judged given its adjacent words and the frequency of its possible Part-Of-Speech tags extracted from a big corpus. The outcome determines the POS tag of each word, distinguishing between proper names and other tags. We decided to abandon this approach due to some problems with the tagger we were using:

- It incurs a very long initialization time which is unacceptable.
- It does not make a difference between true named entities and other types of proper names such as dates and times.
- It is completely useless if names are not capitalized (as could be expected in some email messages, for example)
- The results are only tags of separate words, and do not provide chunking, titles, dependencies or categories.

Creating a database from a corpus This approach gives freedom in determining the criteria of named entities. Each word would have a refined probability (of being a named entity) which includes several factors such as being capitalized, tagged as proper name, appearing in the beginning of the sentence, being adjacent to other proper names or capitalized words. With an appropriate ranking of these factors, even non-capitalized text might be correctly analyzed. Performance seemed to be a lot better, but there was a lot of work involved (practically implementing the NER module from scratch). Eventually, we made the decision to switch to another already existing parser which is fast and efficient:

MINIPAR This dependency based parser is able to identify named entities by employing databases of names and selectional preferences (e.g. a capitalized noun which is an argument for the verb 'say' is likely to be a person). It performs well and provides categories, tags and dependencies between words. It even chunks consecutive names, but it does not tag them as named entities unless it is sure.

No database contains all possible names of persons and organisations, and indeed in many cases, infrequent *names* are not recognized by MINIPAR, but we would still like to link them. Our solution to this was matching information from this tagger and from the infrequent-noun analysis. A chunked sequence of capitalized and infrequent nouns is assumed to be a named entity. Given an unknown name, MINIPAR would chunk it and determine its tag as noun, and then the words are tested to find whether they're infrequent.

2.2.2 How to identify NE types?

MINIPAR already identifies the categories person, location and organisation. Unfortunately, it often fails to specify a category and sometimes it gives the wrong one. It does try to specify all the possible categories of a name, for example by returning both 'location' and 'person' (which we take as category unknown). But on many occasions, it

seems to fall back to 'Person' as some sort of default, making this specific category unreliable. For example, 'Rocky Mountains', 'Wall Street' and 'Peter Stuyvesant Travel', are all considered to be person.

We decided that the category 'person' is too polluted with mistakes to rely on. Therefore, only Locations and Organisations are specified and the rest is unknown. This goes both for entities tagged as 'person' by MINIPAR and infrequent, chunked, capitalized, consecutive nouns.

2.2.3 What context should be given?

To disambiguate the reference of names, especially frequent names, we decided to use MINIPAR's list of *titles*: professor, president, doctor, CEO etc. The problem is that it is not always clear which name in the text the title applies to. For this, we parse the information that MINIPAR provides and build a dependency tree. In some cases, typically fixed titles, such as prince, baroness, doctor, the title governs (is the parent of) the name. On the other hand, positions, such as CEO, chairman or agent, are usually governed *by* the name, whether they appear before it (e.g. president Bush) or after it (e.g. Don Clements, CEO of . . .).

After having experimented extensively with dependency trees, we developed a method that gives reasonable results. Our first approach was to examine the parent as well as the entire subtree of a named entity and extract titles and even other names which are in relation to the entity, but this quickly turned out to be wrong. We realized that it would be quite hard to disambiguate between context that helps to identify a named entity and context which is part of "the story", and would just interfere. For example, in "Bill Gates went to China/bought Intel/met Mr. Bob Roper. . . " any extra context is clearly best avoided. Titles prove as a reasonable approach, and the framework is available and ready for enhancements.

2.2.4 How to find infrequent words?

Infrequent words are a typical problem which can be solved by using the right corpus. For consistency, the best approach would have been using the same database as the tagger uses but that wasn't readable. Instead, we're using a NY Times corpus of 1.4-million words to extract the frequencies. Again, MINIPAR output is utilized here. As mentioned previously the tagger is conservative in identifying named entities. An unknown name will be given a 'person' tag if (and only if) it is an argument of a predicate which is uniquely 'animate' (e.g. say). Otherwise, no tag is assigned. Our database contains words and their frequencies and is used only for nouns that are not already recognized by MINIPAR. Luckily, the tagger provides the root form of a word quite accurately. A word is therefore considered infrequent by our module if the sum of the frequencies of it and its root form is less than 2.

2.2.5 The old-new distinction

Names are repeated in almost any text. The first reference to a name is typically the most elaborate, and subsequent references are shorter (usually using only first or last name). Since we don't want to overgenerate links, it seemed clear that any name should only appear once. As described above, we don't only deal with proper names but also with infrequent nouns, so the added challenge was trying to avoid linking both 'caiman'

and a subsequent ‘caimans’. We employ pattern matching to try to find if a new name already exists in the list. For this we check the name, type and title.

2.3 Overview of our module

The module roughly executes the following steps :

1. Read text (file or standard input). If format is XML, strip XML.
2. Feed text to a sentence splitter. Match the original text to the split text, extract sentences, assign offsets to sentences and words
3. Feed the text to MINIPAR. The output is matched with the original text and information is extracted: offset - bare word - full name - dependency(parent) - syntactic tag - semantic tag.
4. Identify NEs with help of semantic tags. In case the semantic tag is ‘Location’ or ‘Corpname’, add the tag type to the word information.
5. Identify infrequent nouns using syntactic tags and the database of infrequent words
6. Use dependency trees to assign titles to the appropriate names.
7. Test each named entity to the global list of NEs. If it is a new name, add it to the list.
8. Write the original text to a new XML-format, and append infrequent nouns and NEs with offsets and possibly types and titles, output to the Control module.

Example of our module:

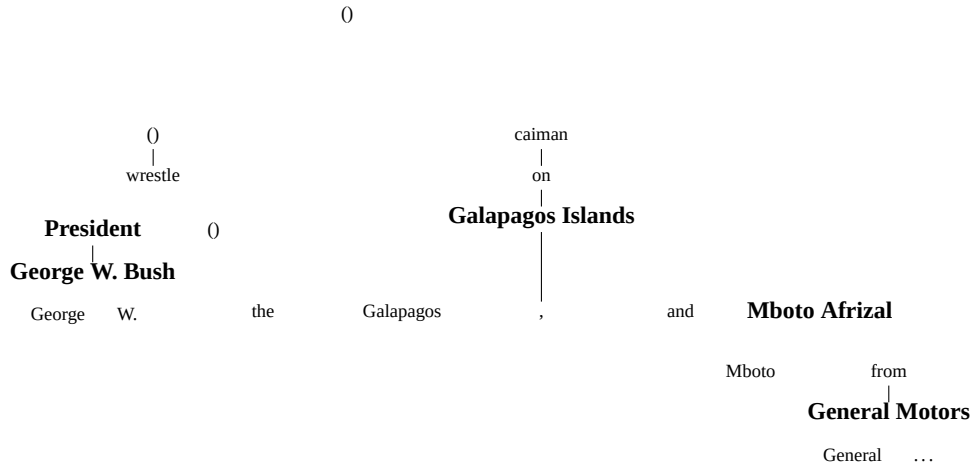
President George W.
Bush wrestled caimans on the Galapagos Islands,
and Mboto Afrizal from General
Motors had to rescue him.

Result from the sentence splitter:

$\langle s \rangle$ President George W. Bush wrestled caimans on the Galapagos Islands, and Mboto Afrizal from General Motors had Sample of the MINIPAR output:

idx	name	root/full name	tag	parent-idx	category
1	President	–	Noun	4	title
3	Bush	George W. Bush	Noun	**1**	person
5	caimans	caiman	Noun	E4	–
9	Islands	Galapagos Islands	Noun	6	location
13	Afrizal	Mboto Afrizal	Noun	9	–
16	Motors	General Motors	Noun	14	corpname

The dependency tree:



The result of our module is:

- ⟨*name*⟩George W. Bush⟨/*name*⟩
  ⟨*type*⟩properName⟨/*type*⟩
  ⟨*subtype*⟩unknown⟨/*subtype*⟩
  ⟨*title*⟩President⟨/*title*⟩
- ⟨*name*⟩caimans⟨/*name*⟩
  ⟨*type*⟩infrequentNoun⟨/*type*⟩
- ⟨*name*⟩Galapagos Islands⟨/*name*⟩
  ⟨*type*⟩properName⟨/*type*⟩
  ⟨*subtype*⟩location⟨/*subtype*⟩
- ⟨*name*⟩Mboto Afrizal⟨/*name*⟩
  ⟨*type*⟩properName⟨/*type*⟩
  ⟨*subtype*⟩unknown⟨/*subtype*⟩
- ⟨*name*⟩General Motors⟨/*name*⟩
  ⟨*type*⟩properName⟨/*type*⟩
  ⟨*subtype*⟩organisation⟨/*subtype*⟩

2.4 Some Implementation details

We implemented our module in Java, and are using two external programs, Sentence-Splitter[2] written in Java, and MINIPAR[3], a broad coverage dependency-based tagger,

[2]http://www.comp.nus.edu.sg/ qiul/NLPTools/JavaRAP.html
[3]http://www.cs.ualberta.ca/ lindek/minipar.htm

which supports several platforms. Java is platform-independent, but MINIPAR isn't. Therefore, we had to support different versions of it and call the right one (determined by a runtime check).

Sentence Splitting This task is not a trivial one, as it requires not only pattern matching but also some sort of database for abbreviations (Corp., Dr. etc.). We decided to use an external Java code, which is easy to call from the within Java. The only intervention we implemented was splitting lines that are separated by a blank line prior to invoking the sentence splitter.

Database of frequencies Our database is computed offline using a big training corpus. Each word is made non-capitalized and added to a hashtable which is eventually written to a file. When the NER module is called, a separate thread is launched to read the file and reconstruct the database.

2.5 Evaluation

2.5.1 Criteria

We evaluate the module mainly by comparing our output with the output that was proposed by the project members. We will check whether words are correctly or incorrectly recognized, and in case of NEs consisting of more words: whether the recognized words were correctly mapped out: exactly those words that are part of the NE should be underlined, no less and no more. Typical problematic cases on this point are Named Entities like 'Secretary of Defense Bob Roper', where the capitalization confuses MINIPAR so that the entity 'Defense Bob Roper' gets linked.

Besides this, we will evaluate whether the extra information about the NEs was correct, and express this as a percentage of the total number of correct links.

2.5.2 Results

The overall results of the NER module are good. We fed it perfect input, assuming that the Control Module had done its part of the job properly. The following table presents our module's performance on a set of texts of different types and formats:

Evaluation	test HTML	test plain	test mail	Justin	Bush	Galapagos	mail	Total
1. Manually	4	11	6	14	19	13	14	81
2. Correct NE	4	7	6	12	19	13	11	72
3. Mismatch NE	0	3	0	2	2	0	0	7
4. Total NE	6	14	9	21	22	19	17	108
5. Have type	0	3	3	3	14	7	5	35
6. Correct type	0	3	3	3	14	7	5	35
7. Have title	0	0	0	2	3	0	1	6
8. Correct title	0	0	0	1	3	0	1	5
Precision	67%	50%	67%	57%	86%	68%	65%	67%
Recall	100%	64%	100%	86%	100%	100%	79%	89%

The table above displays the number of names that were (1) manually linked by our group (2) correct with the manually chosen ones (3) correct but not exact (e.g. 'Defence Bob Roper' for 'Minister of Defence Bob Roper') (4) total number of NEs our module supplied (5) NEs that our module supplied a type for (e.g. location) (6) those whose type was correct (7) NEs that our module supplied a title for (e.g. president) (8) those whose title was correct. The set of test files consists of three tests the teacher supplied and four others chosen by our group. The overall performance of our module is:

Criterion	Performance
Precision	67%
Recall	89%
Information on titles	8%
Correct titles	83%
Information on types	49%
Correct Types	100%
Correct Underlining	90%

Table 2.1: Performance of the NER module on perfect input

The low precision has a clear cause: the corpus we are using for determining the frequency of words is at some points inadequate. First of all because it is not large enough: 1.400.000 words is not enough to distinguish between not-very-frequent, infrequent and typing error. This makes that our module selects words like 'transpostation', 'invaders', 'filmography'. Besides that, the frequency of words in a *newspaper* can differ significantly from language used in e-mail or web sites. Words like 'funk', 'hello', 'bro' (as in "yo bro!") all get linked.

2.6 Improvements and Further Research

Robustness The sentence splitter and especially MINIPAR sometimes act unpredictably. MINIPAR chokes on ampersands, skips seemingly random lines, gives different outputs depending on the platform it is running on, etc. This kind of problems are certainly not unsolvable, but just very time consuming. For this project, we have tried to focus on a module that gives basically interesting results, but the stability of the module is certainly something that would require more attention.

2.6.1 Infrequent words

The method we developed for identifying infrequent words, does work quite well for nouns and adjectives. For verbs however, there are too many inflections that the frequency counter all counts separately. If we would search for infrequent nouns, even a threshold of 1 would be far too high. There are two solutions: one is to take a more adequate corpus, of a least over 5 million words. Another, more sophisticated solution, is to feed the corpus to MINIPAR, to trace all words back to their roots. We would then have far more frequency information to compare our infrequent words with.

2.6.2 More context

A relevant context information is necessary in order to resolve very common proper names. A more refined analysis would try to pick up certain words, dependencies or relations which would be safe to use in order to provide more context. For example, words like ‘of’ (e.g. Bill Gates of Microsoft), and some uses of ‘be’ as the main verb (‘George is the principal of Presley high school’, ‘Mary has been the chairman. . . ’)

Chapter 3

Information Retrieval using Google

3.1 Motivation

Suppose Sam, our average Internet user, is curious about the current president of the United States of America. She wants to find out more about the background of George W. Bush. Being a product of the twenty-first century, she might search the World Wide Web for a biography or maybe his personal website. Chances are she would use Google, the current most popular search engine, which claims to index over eight billion web pages at the time of writing. Sam might type in "George W. Bush" in Google and get back a list of search results. Opening these Google hits in her web browser, she finds that the second search result links to the personal website of George Bush. On this website she finds a biography which satisfies her curiosity.

Maybe Sam's interest in George Bush was sparked when reading an electronic text about the current president of the USA. If the name "George W. Bush" in that text would already have been selected as an entity that she might like more information about, then it would have saved her much effort if it had been automatically linked to the homepage of the current president.

This task is the topic of the current chapter. To describe it more formally: given a list of named entities that have been selected automatically from some input text, retrieve a single link to useful background information using Google. The list might specify whether the named entities are proper names or infrequent nouns, as introduced in Chapter 2.2 on Named Entity Recognition.

3.2 Dedicated websites

We introduce the concept of a *dedicated website* for a proper name. It is supposed to capture our idea of a website that contains useful background information on a proper name and is suitable for linking. A dedicated website for a named entity is a web page that:

- is the homepage for the named entity; or
  gives a description of the named entity; and

- is only about this named entity; and
- is not about the named entity in a specific context.

3.2.1 About Google

As mentioned before, Google currently is the most popular search engine for the World Wide Web. Using Google to select possibly dedicated websites seems an obvious choice. Google guarantees that the words in the query will be present in the search results. This is of course a requirement for finding dedicated websites for persons. Google also has the advantage that it provides a convenient programming interface [4], which makes searching the Internet with custom applications simple.

It is beyond the scope of this paper to explain how Google searches the Internet. However, because we are trusting Google to select possibly relevant links, it is useful to list some of the important features Google uses to analyse websites [3], [2].

- Words in title tag, links, and headings
- Words in bold
- First 25 words of text on a page is given more weight
- Words in URL
- Phrases used to describe pictures
- DMOZ listing (listing in The Open Directory Project [1])
- PageRank (statistic about links on the website and links on other (popular) websites [2])
- Back Links (links from other websites to this website)

Google uses complex methods to decide which pages to show the user and in many cases the first Google hit is the one the user is looking for. For our annotated data set of 52 people (see Sections 3.4.1 and 3.4.6 67% of the first Google hits is dedicated. We will show that our module returns dedicated pages for 76% of these people (see Section 3.4.6).

3.3 Experiments

3.3.1 Domain restrictions

What constitutes a link to relevant background information differs for different types of named entities. For instance, for an obscure city in South America a map showing the city's location might be most useful. For a well-known political leader like George Bush on the other hand, a link to his biography might be most informative.

Because of the limited time available to work on this task, we have restricted our attention to proper names. In particular, we have focused on proper names that are persons.

Feature name	Example
Named entity in Domain	www.GeorgeBush.com
Named entity in Path	www.usa.com/George_W_Bush
Named entity in Title	`<title>`George W. Bush Online`</title>`
Named entity in Keywords	`<meta name='keywords' content='Bush'>`
Named entity in Headings	`<h1>`G. W. Bush`</h1>`
Named entity in Body text	President George W. Bush wrestled ...

Table 3.1: The features used to represent dedicated websites. They are classified based on these features.

3.3.2 Algorithm

For persons we aimed to retrieve a dedicated website, as defined above, using Google. As each search result considered for linking has to be retrieved and processed, considering too many search results would be very slow. Therefore we have decided to use only the top ten Google search results.

The following algorithm follows naturally from the restrictions in the preceding paragraphs:

- Get top ten Google search results
- For persons, select the first dedicated website in results if one is present
- Otherwise, fall back to the first Google search result

Notice that it decides between multiple dedicated websites by selecting the one ranked highest by Google.

3.3.3 Prior knowledge

Given a website and a named entity, how do we automatically classify the website as dedicated to this named entity or not? We can already think of some properties of websites that we suspect are highly correlated to whether a website is dedicated. Suppose, for instance, that we have to decide whether http://www.georgebush.com/ is dedicated to George W. Bush. Then the fact that the words *George* and *Bush* are in the URL of the website, provides a strong indication that it actually is dedicated to George Bush. In general, the presence of the name of the named entity in the URL provides a strong indication that the website is dedicated to that named entity. Ideally, we would like to supply this *prior knowledge* about informative characteristics to our classification algorithm.

3.3.4 Abstract representation

In order to supply our classifier with informative characteristics, we handed it an abstract representation of websites. A website was described using a fixed set of features, that each represented what we though were useful characteristics for classification. Table 3.1 shows the features we have used.

3.3.5 Binary versus continuous features

Having decided on using abstractions of the websites using features, we should decide whether to use binary or continuous features. E.g., for a binary feature the name of a named entity is either present in a URL or it is not. For a continuous feature, however, the name can be present to some degree. Part of *George Bush*, for instance, is present in www.bush.com/. Continuous features can represent more fine grained information, but require a bigger train set (see below). For simplicity and to reduce the amount of manual classification work, we have used binary features.

3.3.6 Two classifiers

For our experiments we have tried two classification algorithms: a *perceptron* classifier and a *majority vote* classifier. The perceptron tries to deduce a linear separation of the data into two classes, while the majority vote classifier uses a vote among at least three nearest neighbours of a new example in the train set to classify the example.

3.3.7 Perceptron

A perceptron uses a linear discriminant function to separate the data into two classes. It classifies according to the rule:

$$\begin{aligned} \sum_i w_i \cdot f_i \geq \theta \quad &\rightarrow \quad +1 \text{ (dedicated)} \\ else \quad &\rightarrow \quad -1 \text{ (non-dedicated)} \end{aligned}$$

where each feature f_i is either 0 or 1 (absent or present) and each w_i represents the relative weight of feature i. θ sets a threshold to the evidence the perceptron needs to see before classifying a website as dedicated.

To find the weights w_i for the perceptron that minimize its *Summed Squared Error* over a train set D, the well-known *delta rule* can be used iteratively until convergence.

3.3.8 Majority vote

A vector of n binary features can only take on 2^n possible values. Because we are using a small number of binary features, there is also a small number of possible values the feature vectors can take on. Therefore many samples (websites) will be projected onto the same feature vector. In fact, when a new sample needs to be classified, usually the classifier has already seen train samples with the same feature vector.

The majority vote classifier we have designed is a modified Nearest Neighbour classifier that is able to handle this case well.

When multiple samples are projected onto the same point in the feature space, the majority vote assigns new samples to the same class of the majority of the samples at that particular point in feature space.

Ideally, there will be a sufficient number of training samples at every point in feature space. However, when this is not the case, we need to come up with a reasonable classification as well. So we look for all nearest neighbours (according to Euclidean distance) and assign the new sample to the class of the majority of nearest neighbours. Because the distance between samples can also be zero, this method also works for

Rank of Google search results included	Performance
No	69.9%
Yes	69.4%

Table 3.2: Performance of the majority vote classifier on the manually annotated data set, averaged over 100 runs

the case when there is a sufficient amount of training samples at some point in feature space.

For binary features (0 or 1), the sum of the features is equal to the Euclidean distance squared. So it is monotonic in the Euclidean distance. For ease of implementation, we just use the sum of the features as a distance measure, as the results should be equivalent to using the Euclidean distance.

3.4 Evaluation

3.4.1 Manually annotated data set

To get a grip on dedicated websites for persons, we have created a list of 52 people likely to have one or more websites dedicated to them. For these people we have manually classified the first 10 Google search results as either dedicated or not dedicated, resulting in a data set of 520 manually classified websites.

3.4.2 Experiment 1: Evaluation majority vote classifier

Averaged over 100 test runs, trained with 90% of the annotated samples (see Section 3.4.6) and tested with a random test set (10% of the annotated samples), the majority vote classifier achieves a performance of 70% (see table 3.2).

We tried to add the ranking of the search results as a feature, making it one when the sample belonged to the top five of hits and zero otherwise, but since this did not increase performance, we did not include it in our final version of the classifier. However, we do use the Google ranking of the search results implicitly, because the websites are classified one after the other based on their ranking. First the first Google hit is classified, after that the second, etc. As soon as a dedicated website is found it is returned and the remaining search results are discarded.

3.4.3 Experiment 2: Evaluation perceptron classifier

We evaluated the performance of our implementation of a perceptron classifier on the manually annotated data set described above. We randomly divided the data set in a train and a test set for each run. The train set always contained 90% of the examples, while the test set contained the remaining 10%. As training a perceptron is much slower than training a majority vote classifier, we only evaluated its performance by averaging over 20 runs (compared to 100 runs for the majority vote classifier).

Just like for the majority vote classifier, we tried adding the rank of the websites in a Google search as a feature. The results are shown in Table 3.3.

Compared to the majority vote classifier, the perceptron performs much poorer. Apparently a linear separation of the feature space is not able to discriminate well between dedicated and non-dedicated websites. Adding the Google rank of the website

Rank of Google search results included	Performance
No	60%
Yes	62%

Table 3.3: Performance of the perceptron classifier on the manually annotated data set, averaged over 20 runs

IRGoo class/True class	High	Medium	Low
High	76%	15%	8%
Low	0%	0%	0%

Table 3.4: Link quality of the dedicated websites IRGoo returned for the annotated set

does increase the performance slightly. As the perceptron performs poorer than just using the first Google search result, which is a dedicated website in 67% of the cases, it is not surprising that information about whether a website has a top five ranking by Google, improves its results.

3.4.4 Noteworthy feature vectors

In our annotated set, we found 37 different feature vectors, that is about half of the $2^6 = 64$ possible feature vectors (see table 3.1 for the features we used for classification). There were relatively large amounts of samples for seven of the feature vectors we found. The size of these groups ranged from 21 to 55 samples. In most cases, the amount of dedicated samples in these groups was slightly larger, so new samples with the same features were classified as dedicated, although they could also easily be non-dedicated. This accounts for at least some of the inaccurate results.

There were 13 samples with only the 'Named entity in Path'-feature of which 12 were dedicated (which is rather strange when you think of it). For comparison: there were three samples with only the 'Named entity in Domain'-feature, of which only one was dedicated (of course 'Named entity in Domain' will often occur in combination with other features, so it is still a useful feature, it just does not occur often on its own).

3.4.5 Classifier choice

The majority vote classifier outperforms the perceptron classifier by 10%. The majority vote classifier is therefore to be preferred for detecting dedicated websites in the algorithm of Section 3.3.2.

3.4.6 Experiment 3: Module performance

Annotated data set

For our manually annotated set of 52 people, IRGoo always returned a link with high confidence. This gives a recall of 100%. Of these links 76% actually is high quality. In 15% of the cases the websites contained information about the right person. Only four websites contained no information about the right person at all. Table 3.4 gives an overview of link quality for the results of our annotated set.

Classifier/We	High	Medium	Low
High	70%	24%	6%
Low	0%	0%	0%

Table 3.5: Link quality of the dedicated websites IRGoo returned for the 17 persons in the test set

Test texts

There were 17 person names present in the test texts. The complete texts can be found in Chapter 7. We assumed perfect output from the NER module, with all person names found and recognized as persons.

Notice that IRGoo returns a link with high confidence for each person, recall is again 100%. We evaluated a link as high when the website was a dedicated website for someone with the name we were looking for, even though it could be a dedicated website for another person with the same name. This probably happened for some of the names in the test e-mail, which contained names of businessmen.

Table 3.5 gives an overview of the results. The precision is slightly lower than for the annotated set. This is because the test set contains some names of less well known people like Bill Black, Sam the Sham and Kevin Kane. We assume that a dedicated website exists for every person. Of course, there will be always people without a dedicated website.

3.5 Future work

There are several ways to extend the IRGoo module. A very obvious option for improvement is to extend the types of words it can handle. So far, we have restricted ourselves to the subcategory person of proper names. This should be extended to include the other subcategories, location and organisation, as well. Perhaps it is even possible to find dedicated websites for proper names of subcategory unknown. For every new category, we will have to find a proper abstraction of the type of websites we want to look for. For websites of subtype organization this could be a corporate website, e.g. www.gm.com for General Motors. Of course, the next step will be to allow for different categories of named entities, like infrequent nouns.

The majority vote classifier we trained gives acceptable performance, but we feel this could be improved. The performance of classification of websites can be improved in two different ways. First, we could have a look at the features we extracted to characterize a website. We discovered some very common feature vectors for dedicated pages as well as non-dedicated pages. Perhaps some other features are better for recognizing dedicated websites. It is also possible to have the classifier identify interesting features automatically.

The second way to improve the classification results is to implement new classifiers with better properties to distinguish between dedicated and non-dedicated websites. We could train a non-linear discriminant. Non-linear discriminants can model boundaries between classes that are nonlinear. This could yield better results for our case, for example in the above mentioned case of the URL www.bush.com, in which only a part of the name George W. Bush is present.

Other possible extensions consider the information retrieval part. We could try to improve the information retrieval by incorporating context in the query. At this

moment, our module returns dedicated websites for person names and not necessarily for right persons. We could try to distinguish between different person with the same name. Context can also be useful for other subtypes of proper names. E.g. when we are looking for an corporation named General Motors, it could be useful to extend the query with the words 'corporal site'. Sometimes it will not be possible to extract extra information from the context (e.g. for small texts). For these cases other strategies can be applied, like (automatic) query expansion or refinement.

Fans of president Bush will like the link www.ilovegeorgebush.com, but perhaps members of the Democratic Party will prefer more objective information about George Bush. So a useful link for one person does not have to be a useful link for someone else. Preferences of users could be stored in user profiles. IRGoo could ask users for feedback on proposed links.

3.6 Technical description

The input of the IRGoo module consists of the output of the NER module. It is an XML-file with named entities and some meta-information for these named entities.

The IRGoo module classifies Google hits with the majority vote classifier and returns the first dedicated website found. In all other cases it falls back to the first Google hit. For every returned link, it adds a confidence measure (high or low). When Google returns no hits, it returns no link for this particular named entity. IRGoo also returns the Google hit count for each named entity.

3.6.1 Handling special cases

So far we have assumed that the IRGoo module always returns a link to a dedicated website. Of course there are a number of special cases where this is not possible. The first case arises when our module cannot find a dedicated link within the first 10 Google hits for a named entity. We solve this problem by introducing a confidence measure for the relevance of links. When our module returns a dedicated link, the confidence parameter is set to *high*. If our module classifies all links as 'non-dedicated', the first Google hit is returned, and the confidence is set to *low*.

Google searches the Internet, so it is possible that the source URL of the input document or hits from Wikipedia are returned. Wikipedia is the domain of the IRWiki module (see Chapter 4), so we ignore these links. Ignoring means that we detect the link and then stop the feature extraction and classification process for this specific link. We also ignore links to the input document, because it is very annoying to have a link to the document you are reading when you want extra information about something that is mentioned in the same document.

The third and last special case arises when Google does not return any links at all. Our module will ignore this named entity, return nothing and proceed with the next named entity (or stop when it is the last).

3.6.2 Sample output

```
<?xml version="1.0" ?>
<ltp xmlns="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
xmlns:xsi="test" xsi:schemaLocation=
"http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome irgoo-to-control.xsd">
```

```
<namedEntity length="14" offset="11">
        <name>George W. Bush</name>
        <type>properName</type>
<subtype>person</subtype>
        <title>President</title>
        <gooURL confidence="high">http://www.bushorchimp.com/</gooURL>
        <googleHitcount>2820000</googleHitcount>
</namedEntity>

<namedEntity length="7" offset="35">
        <name>caimans</name>
        <type>infrequentNoun</type>
        <gooURL confidence="low">
http://www.flmnh.ufl.edu/natsci/herpetology/brittoncrocs/cnhc.html
</gooURL>
        <googleHitcount>7600</googleHitcount>
</namedEntity>

<namedEntity length="18" offset="50">
        <name>Galapagos Islands</name>
<type>properName</type>
<subtype>location</subtype>
        <gooURL confidence="low">http://www.darwinfoundation.org/</gooURL>
        <googleHitcount>122000</googleHitcount>
</namedEntity>

<namedEntity length="13" offset="69">
        <name>Mboto Afrizal</name>
        <type>properName</type>
<subtype>person</subtype>
        <googleHitcount>0</googleHitcount>
</namedEntity>

<namedEntity length="14" offset="90">
        <name>General Motors</name>
        <type>properName</type>
        <subtype>organisation</subtype>
        <gooURL confidence="high">http://www.gm.com/</gooURL>
        <googleHitcount>2600000</googleHitcount>
</namedEntity>

</ltp>
```

Chapter 4

Information Retrieval using Wikipedia

Peter van der Meer & Paul Manchego

4.1 Motivation

Relevant sources of information to link to include encyclopedias and dictionary type web servers. Wikipedia [5] is a well known online encyclopedia. It is designed to be read and edited by anyone, with editions of varying sizes in 190 languages. Its the world's largest encyclopedia, containing 460,000 articles (growing 3000 per day) and 77 million words. Wiktionary [6] is based on the same concept, but provides dictionary information. The Wikipedia module, also called IRWiki tries to provide the best link given a Named Entity.

This chapter describes the approach of the IRWiki module; how can links be retrieved, what kind of information is needed for it and and what choices are made. Next the performance of the system is evaluated. Pitfalls are identified, and solutions to them are presented. This chapter concludes with suggestions for future research.

4.2 Approaches

This section lists a couple of approaches that can be used to obtain relevant URLs for a Named Entity (NE). The approach used for this project is mentioned last. For NEs, a distinction is made between Proper Nouns and Infrequent Nouns. If a NE is a Proper Noun, the most relevant Wikipedia URL is sought, and a confidence measure of this URL is provided. If an NE is an Infrequent Noun, a Wiktionary link is sought, and in addition a small definition of the word is provided. The language domain is restricted to English. No context related to the NE is taken into account when trying to retrieve the most relevant link.

4.2.1 Retrieval using Wikipedia's online search engine

The online search engine can be used to make queries using a Named Entity as input. The resulting page then needs HTML-parsing to obtain a collection of URLs. If there

is more than one URL, the best one has to be selected according to some link evaluation criteria. The advantage to this approach is that the most recent information is accessed, as Wikipedia or Wiktionary is subject to changes. However, there are a couple of disadvantages to this method:

- Accessing the Wikipedia and Wiktionary site is relatively slow.
- A search resulting in an HTML page needs to be parsed to obtain possible URLs. The HTML format of the pages differs from time to time, which makes this a non-trivial problem. Sometimes the resulting page just refers to the Google and Yahoo search engine, if the web-server has gotten too many requests.
- The search engine only allows simple queries.

4.2.2 Retrieval using Wikipedia and Wiktionary's database

The Wikipedia database is also available for download. The database is in SQL format and can be imported into a database server such as mySQL. Once an SQL server is set up, queries can be made using any programming language that supports interaction with that server. The database can be searched for the most relevant URL, and a confidence measure can be determined. There are a couple of advantages to this method:

- It can perform queries much faster than the web based search engine.
- Queries can be customized, which allows more options than the online search engine

There are also some disadvantages:

- The Wikipedia database is quite large. For English it is 29GB, thus, a capable server is required. The Wiktionary database for English is only 142MB.
- As the Wikipedia domain is changed all the time, the database will need updates periodically.
- The available faculty computers are not powerful enough to host such a database.

4.2.3 Retrieval using Google API

The Google search engine allows queries to be limited to a specific domain (e.g. en.wikipedia.org). There is a Google Application Program Interface (API) available for Java, which can be used to retrieve URLs for a Named Entity. Currently, the highest ranked Google result is returned. Even though Google is one of the best search engines around, the highest ranked result is not always a good link, so a confidence measure is determined for each Wikipedia link. The URL confidence measure is always one of the three values 'low', 'medium', or 'high'. In order to determine the confidence for a given page and NE, all HTML tags are stripped from the HTML page, and the frequency of the NE is counted. If the NE occurs 10 times or more on the page, the URL confidence is considered 'high'. If it occurs between 5 and 9 times, the URL confidence is considered 'medium'. If the NE occurs less than 5 times on the page, the confidence is 'low'. In a relevant Wikipedia page a Named Entity or part of it occurs at least 15 times or more. Links with average confidence are uncommon; experimentally only one was encountered, for the NE "KCAL". The document was about the calorific

value of coal and kcal/kg occurred seven times. For the Wiktionary the relevance for the highest ranked Google hit is considered ‘high’. There are a couple of advantages to this approach:

- Google is a more advanced search engine than Wikipedia’s own search engine. Google takes user behavior and other methods into account when providing the most relevant link, which increases the chance for the most relevant page to be found directly.
- The Java API is freely available for download, whereas there is no API for Wikipedia.

This approach also has some disadvantages:

- Queries will be performed slower, compared to the database approach.
- API requires a search key that is linked to an email-address. The search key has a limit of 1000 queries per day. This can become a problem if the system is used extensively.
- The Google API returns a pruned and ranked list of URLs. In some cases a correct URL might not be present in this result set. One of the reasons might be that Wikipedia is constantly updated and Google has not had time to re-cache the page yet.

4.2.4 Retrieval using link construction

Wikipedia URLs and pages have a very specific format. The Named Entity “George W. Bush” can be translated into the following link: http://en.wikipedia.org/wiki/George_W._Bush. The first part of an URL is always the same. Each word in the last part of the link is capitalized and spaces have been replaced by underscores. To validate if the link is valid the `<title>` tag of the HTML document is checked for the NE. If a valid link is found, the confidence is considered “high”. If the Named Entity is an Infrequent Noun a small definition is also retrieved. The advantage of this method is that it does not require the Google API and easy to implement. The disadvantage is you are dependent on the speed of the Wikipedia web server, which at times is quite slow.

4.2.5 Retrieval using link construction and Google

This is the method we used for URL retrieval. For both Proper Nouns and Infrequent Nouns the method is the same. First the link construction is tried. If no result is found the Google based method is used. If both approaches fail no URL or definition is returned. There are a couple of advantages

- Both methods are easy to combine
- Since less Google queries are made, the Google search key can be used longer.

The disadvantage is that this method is still dependent on the speed of the Wiki and Google web server(s).

Confidence	High	Medium	Low
Proper Noun			
Precision	1.00	0.00	1.00
Recall	0.96	0.00	1.00
tp	26	0	3
Infrequent Noun			
Precision	1.00	0.00	0.00
Recall	1.00	0.00	0.00
tp	15	0	0

Table 4.1: IRWiki precision and recall measurements

4.3 Evaluation

The evaluation measures used were precision and recall. Precision is the fraction of retrieved links that are relevant and recall is the fraction of relevant links that are retrieved. Proper Nouns and Infrequent Nouns and each confidence level are evaluated separately. Table 4.1 shows the results. The test text contained several genres: news about music artists, general news, encyclopedia and email.

Tp is the amount of retrieved links that are relevant. As can be seen, there are very few links with a low and medium confidence level. There are no low or medium links for Infrequent Nouns, but that is due to the implementation. A larger test text will improve the precision and recall measurements in the low and medium region of Infrequent Nouns. The IRWiki module performs quite well with regard to recall and precision. Also, 93% of the returned linkes are very informative.

4.4 Error Analysis

Currently, no context is taken into account. For the NE ‘Bush’ for example, it would be nice to know president George W. Bush is meant, and not a shrub. The link expansion method will fail here if only ‘Bush’ is used. In case a Named Entity is a person, it will be much better to skip the link expansion method and start with Google. The Google method is not always perfect, so it could me modified such that it returns the first link which contains the last part of a person’s name. The NER precision and recall of determining if a Named Entity is a person is not good, so this might introduce another systematic fault. In case of Infrequent Nouns, qualifying the first Google hit with high confidence can be improved.

4.5 Future Research

4.5.1 Improvement of the URL selection and confidence evaluation

Confidence evaluation works well in most cases, but has not been tested thoroughly. By default returning a Wiktionary URL for a Infrequent Noun is not always wanted. Wikipedia can provide more relevant links at times. For example, the word ‘obelisk’ is not present in the Wiktionary but Wikipedia contains quite a nice page.

4.5.2 Improvement of the URL retrieval using context information

In some cases it is possible to use context in the corpus relating to a Named Entity to improve searches. Currently the NER module is responsible for finding contexts, but pattern matching can also be tried, e.g. using prepositions like 'to' that relate nouns to each other to find relevant context.

4.5.3 Examine database approach

The system performance can be improved a lot by using only the database approach, as this is not dependent on the speed of the Wiki and Google servers. If the precision and recall is comparable this might be a good alternative choice.

4.6 Technical Description

Java Development Kit 1.4.2 was used as a programming language. For communication the module uses standard input and output (STDIN/STDOUT). A corpus can contain many named entries. Retrieving an URL, confidence measure and, if needed, a description for each named entity can take a lot of time. Multithreading is employed to accelerate the overall performance. Named entities are assigned to specific threads to accelerate processing of URL requests. As the Wiki servers are slow, it could be faster to use the Google cache when determining link relevance. Unfortunately, the Java API does not support this functionality. Extended markup language (XML) is used as data exchange protocol. Some example input and output for Named Entities is shown below. For each NE, the `<name>` tag is needed to get the name. The `<type>` tag is used to differentiate between Proper Name and Infrequent Noun. Currently context information from the *title* tag is not taken into account as is the `subtype`.

Example NE input:

```
<namedEntity offset="1" length="5">
  <name>George W. Bush</name>
  <type>properName</type>
  <subtype>person</subtype>
  <title>president</title>
</namedEntity>
<namedEntity offset="1263" length="6">
  <name>bush</name>
  <type>infrequentNoun</type>
</namedEntity>
```

Example NE output:

```
<namedEntity offset="1" length="5">
  <name>George W. Bush</name>
  <type>properName</type>
  <subtype>person</subtype>
  <wikiURL confidence="high">wikipedia.org/George\_W.\_Bush</wikiURL>
</namedEntity>
<namedEntity offset="1263" length="6">
  <name>bush</name>
  <type>infrequentNoun</type>
```

```
  <wikiURL confidence="high"> http://en.wiktionary.org/wiki/Bush </wikiURL>
  <description> A horticultural rather than strictly botanical category of
    woody plant that is distinguished from a tree by its multiple stems and
    lower height; usually less than 6m tall. </description>
</namedEntity>
```

Chapter 5

The control module and the user interface

Joeri Honnef & Daan Vreeswijk

This chapter describes two different (but related in the sense that they were created by the same group and have some overlap in their tasks) parts of the project: the user interface and the control module.

The main task of the control module was to facilitate the communication between the different modules. The text that was to be linked had to be passed to the NER module first, and its results had to be sent to the IRGoo and IRWiki modules. Finally, the links had to be inserted in the text at the correct places.

Two different interfaces were created: for e-mail linking and for linking simple texts. The e-mail linker can be integrated in a mail client. The user can choose to pipe every incoming or outgoing message, or maybe only some messages, through the e-mail linker in order to automatically generate links. If a user simply has a text that he wants to be linked, he can use the web interface to enter a text and get the result in his browser, or he can enter a URL and view the results.

5.1 The control module

The bulk of the system is comprised of three modules: the NER module, the IRGoo module and the IRWiki module. Each module performs a series of tasks, with specific input and output requirements. In order to let each part perform their task at the correct time and with the correct input, the control module was written. It was designed to make sure that input, output and errors were handled correctly.

5.1.1 Step by step

The control module accepts text from standard input. It reads in this text, and then runs an HTML-stripper, to be sure that when the source is a web page, all tags are removed[1].

When the text is stripped bare of any unwanted information, the control module puts the text in an XML-schema, starts the NER module, and writes the text to its

[1] Note that with malformed HTML an error might occur, leading to not all HTML being correctly removed and undetermined behaviour of the other modules

input. It then waits for the NER module to finish, and catches it output. If indeed the NER module outputs anything (if no named entities are found, the NER module returns an empty string, in which case the output of the control module is simply the message: "The NER module didn't find anything."), this output is then sent to both IR modules. These are started in parallel, in order to speed up the system.

The control module also creates a timestamp every time a module starts and stops, and calculates the execution time of each module. This can be used to evaluate the modules with respect to speed.

The final task of the control module is to recreate the original text, now with the links integrated in it. For that, it first analyses the output of the IR modules. Both modules give a confidence measure to their URL with the possible values of 'low', 'medium' and 'high'. To choose which link is best for the given named entity, the control module simply picks the link with the highest confidence measure. Because Wikipedia is a user maintained encyclopedia, as opposed to the relative arbitrary websites that can be found with Google, the Wikipedia link is used if both confidence measures are the same. When this is finished, it attaches the results of the time measurements to the text and prints the result to standard output.

5.1.2 Technical description

The control module is written in Python. This was done because of the easy methods Python has for starting and stopping other programs and managing threads. All modules read their input from standard input, write results to standard output and errors to standard error. Every time the control module starts another module, it creates pipes to standard input, -output and -error. This enables the control module to write to the other module's standard input and read from its standard output and error.

In order to insert the links within the text, a function is written that simply writes the original input text to an output string one character at the time. It keeps track of how many characters it has already written, and whenever it reaches a point where a named entity begins[2] the best link is chosen and inserted in the text.

Also, the text is placed in a `div`, to make the integration with the web page simpler. This is against the goal of separating the control module from the web interface, but since it causes no problems if the control module is used to link e-mails, and it is much easier to separate the timing information from the original text if there are divs separating them, it was decided to do this.

Analyzing e-mail causes a bit more problems than analyzing plain text or HTML, since e-mail has specific e-mail headers, that should be separated from the body, possibly adapted to HTML-mail, and then put back into the final result. For this, the existing email-module from python was used. However, due to time constraints the e-mail analysis was not integrated very good into the rest of the control module, so in the end it was decided to create a separate control module for the e-mail annotation. This only differs from the normal control module in that it performs the e-mail analysis at the beginning, and recreates the e-mail at the end.

5.1.3 Evaluation

The control module does its job, and does it fairly well. It starts each module in time and with the correct input (insofar as it has influence on that) and outputs the text with

[2] Remember that the NER module provides offsets and lengths for each named entity

links added at the correct places. So in that respect, it succeeded its design requirements.

The most important part where the control module did not meet its design was in error handling. If one of the modules crashes, the control module is simply at a loss and doesn't output anything. The only error that is handled correctly is if one of the modules does not output anything: then a good error message is outputted.

It is also a pity that there are now two different versions of the control modules, one for e-mail and one for plain texts. However, this does not cause any serious problems: the web interface simply calls the normal control module, and an e-mail client can call the e-mail control unit.

5.1.4 Future work

The two most obvious future works are solving the problems mentioned in section 5.1.3. Errors should be handled more informatively and correctly. So for each module, some exception handling should be built in, and informative error messages should be written to the screen. The two control modules should be integrated into one, with for example a command line argument specifying whether an e-mail or another text type is given as input.

5.2 The interface

5.2.1 Linking e-mails

In order to link an e-mail, a user has to enter an option into his e-mail program. Of course this differs from program to program. An example of such an option can be seen in figure 5.1.

Here, the filter is set so that every message with the word 'autolink' in its subject is sent through the control module, and thus automatically linked. For a more detailed description on how to create a KMail filter one should read the tutorial included in the package download of the email module.

5.2.2 Linking a web page or text

The most standard method of using the linker is via the web interface. Here, the user can enter either a URL or a piece of text, and the web interface then outputs the linked text to the screen. It works by simply calling the control module, passing it the entered text and, after adding some HTML tags to it, writing the output the screen. The web page can be seen in figure 5.2, and a linked text is shown in 5.3.

5.2.3 Technical description

About the e-mail interface, not much can be said. When a user has set an option in his e-mail program, the control module is simply invoked.

The web interface is written in PHP, mainly because this was supported on the available server. It reads in a text or URL (in the case of a URL, the contents of the URL are read into a string), starts the control module, and writes the text to the control module's input pipe. It then waits until the control module is finished, captures its output, adds some HTML-tags to it and writes it back to the screen.

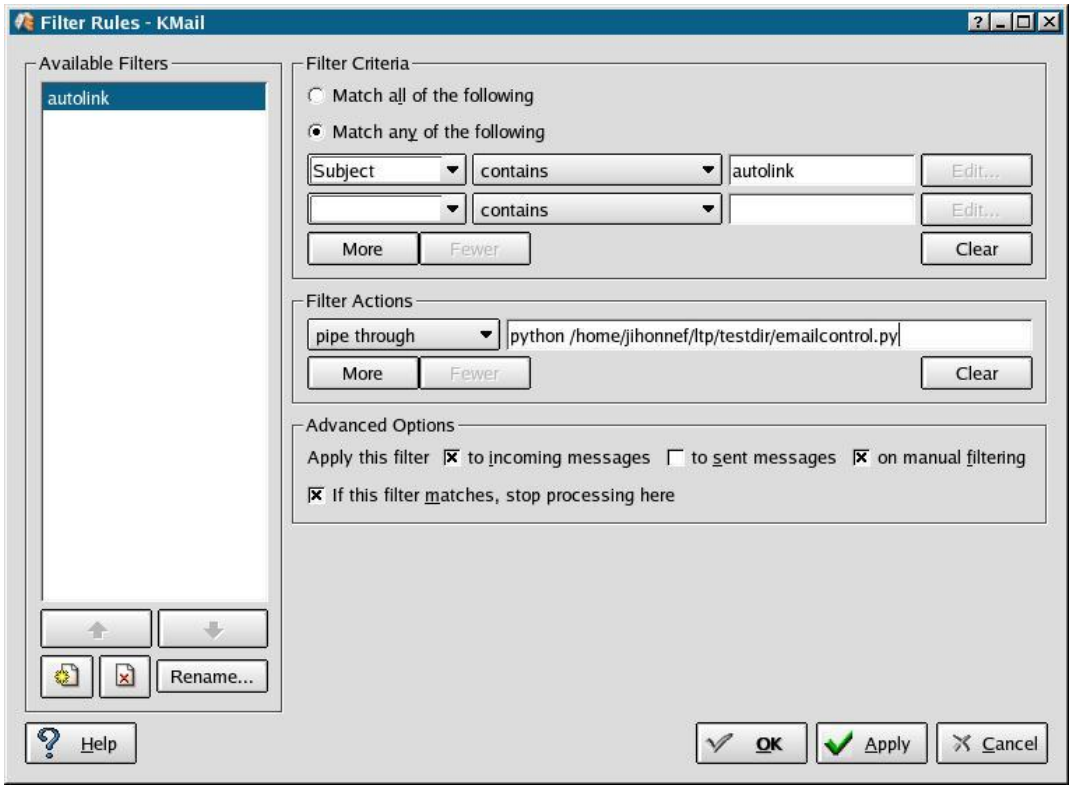


Figure 5.1: Screenshot of the K-Mail filter page

The website is written to be XHTML1.0 and CSS compliant. Because of the wild variety in browsers that all implement the standards a bit differently, this is by no means a guarantee that it will work in every browser, however Firefox, Mozilla and Internet Explorer all seem to work fine. And indeed if a browser cannot display the output, complying to the standards means that the browser then can be blamed.

5.2.4 Evaluation

The interfaces both seem to work well. They were tested with texts of various sizes, and did not cause any problems. A small explanation text is written on the web page, in order to provide a little background on the purpose of the whole program. It is a simple design, with enough options to run the program completely.

Future Work

As of yet, the entire program is callable online through the system's homepage. This means that also for example the NER module could be called directly. This is not very elegant, and all modules should actually be moved to a non-directly accessible directory.

Also, no check on the input text is performed as of yet. So files that can be read through the web interface, can also be parsed through the web interface. This has the

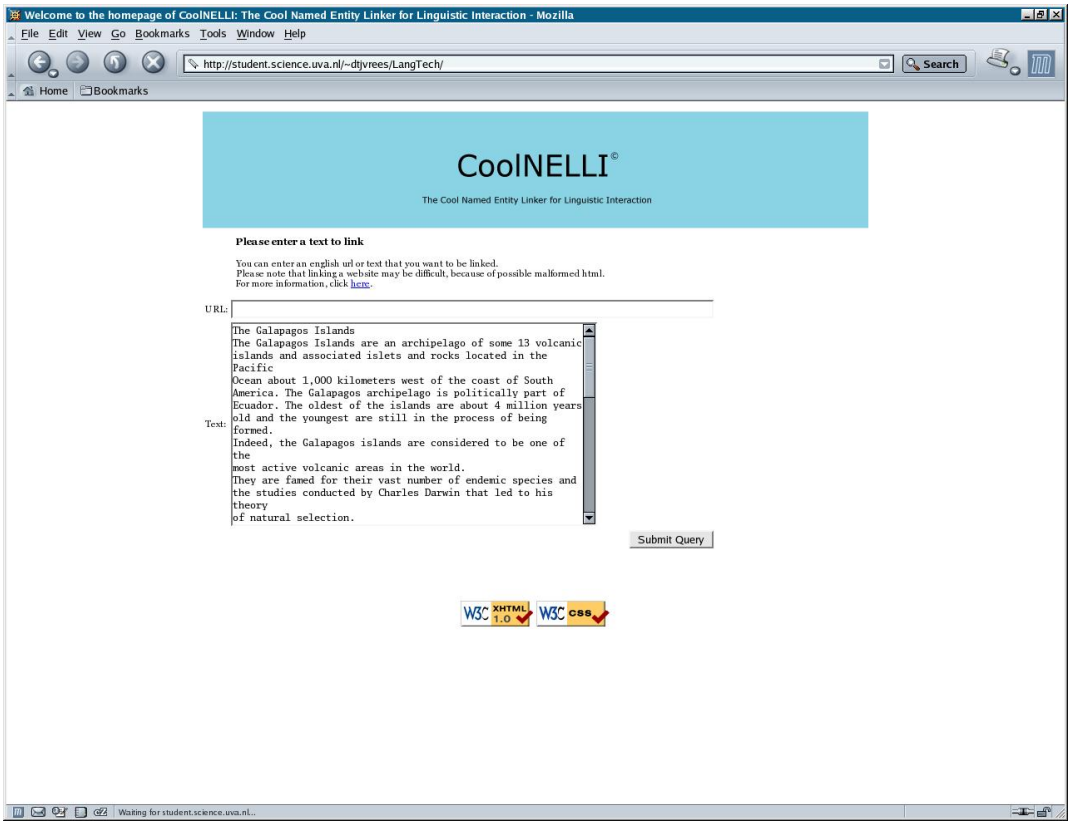


Figure 5.2: CoolNELLI web interface input form

effect that a file like `run`, which is part of the NER module and contains a simple Java-command, can be parsed. This is also not very correct, and for security reasons the website should be modified in order to prevent this.

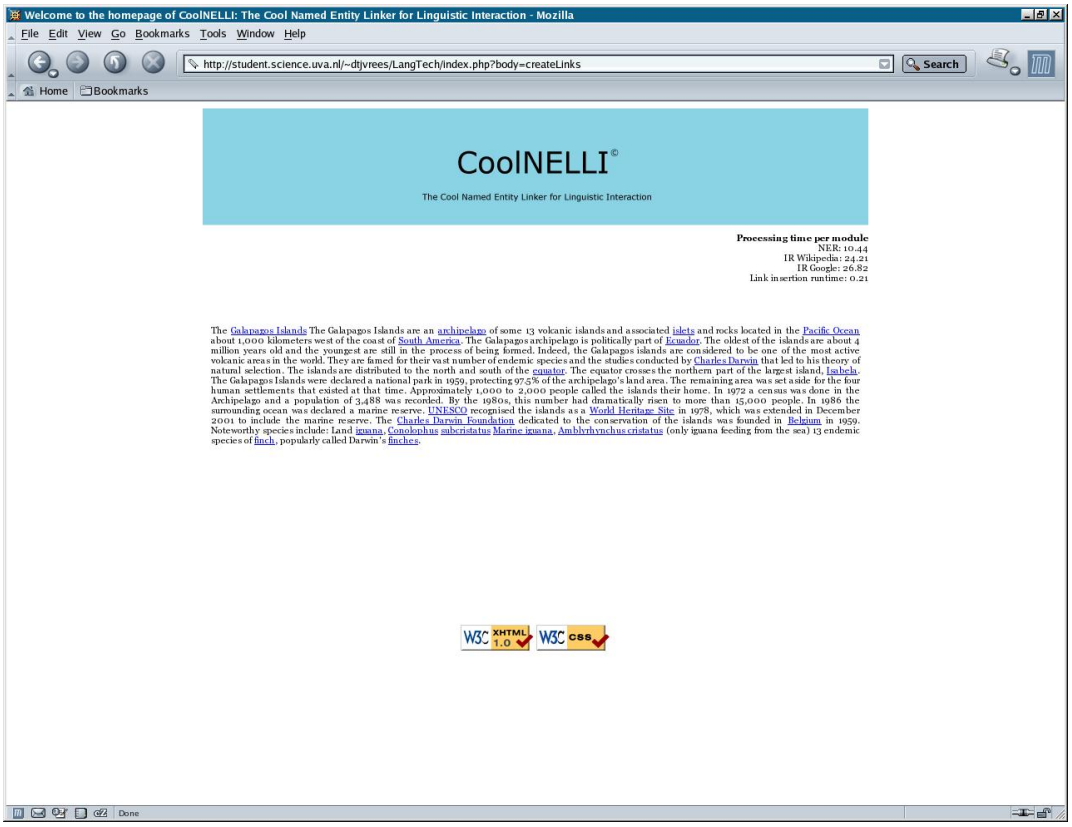


Figure 5.3: CoolNELLI web interface output

Chapter 6

Evaluation

In the preceding chapters each module has been separately explained and evaluated. But Sam, the average Internet user, is not interested in modules, she is only interested in the performance of CoolNELLI as a whole. The performance of CoolNELLI is not just the sum of the results of the modules, because for example an incorrectly marked named entity, say: ‘General’ in stead of ‘General Motors’, will also lead to an incorrect link. These accumulating mistakes are not reflected in the separate evaluations per module. So, now the time has come to evaluate the CoolNELLI system as a whole.

To test the performance of CoolNELLI, a test set of manually linked text was created. The test text, with the proposed words to link marked, can be found in Chapter 7. A total of 57 links are suggested, 30 of which are links to Wikipedia and 27 are links found with Google.

6.1 Linked named entities

Of the 57 proposed named entities to link, CoolNELLI was able to link a total of 55. This gives a recall of 96%, which is very good. However, there were minor problems with some of the links, e.g. we proposed to link ’Global Moment in Time’ and in the output only ’Global Moment’ got a link. We decided to ignore these minor mistakes and added them to the correctly discovered links.

The total amount of links returned by the system was 86, so 31 additional links were found. Most of the time, the system linked infrequent words like ’funk’, ’airplay’, ’bluesy’, ’Conolophus’ and ’subcristatus’. Results about the number of linked named entities are shown in table 6.1.

Wanted	57
Wanted and found	55
Additional	31
Recall	96%

Table 6.1: Recall on test text of Chapter 7

Source	Correct	Incorrect	Recall
IRWiki	47	28	62 %
IRGoo	6	5	55 %

Table 6.2: Numbers of correct and incorrect links generated by IRWiki and IRGoo

6.2 Quality of the links

In this section the quality of the found links is evaluated. For links to Wikipedia a good link is an exact Wikipedia match. For links suggested by IRGoo a good link is a link to a site that contains useful information about the named entity. This distinction has been made because it is easier to find something useful in Wikipedia than when searching the entire Internet. The results are given in table 6.2.

The total number of links generated by IRGoo is, compared to the number of links generated by IRWiki, very low and the recall for IRGoo is also lower. This can be explained by the fact that the IRGoo module focused on person names and that in the result most links to persons are linked by IRWiki.

The overall recall is calculated with the following formula:

$$\frac{\text{correct IRWiki links} + \text{correct IRGoo links}}{\text{number of found links}}$$

This gives an overall recall of 62%, which is quite acceptable.

6.3 Speed

The execution time through the web interface is at least 30 seconds. It usually even takes about a minute to process a text, extract named entities, find appropriate links and insert them into the document. An average user is not prepared to wait a minute before she gets results. So the current version of CoolNELLI available online is not very convenient.

However, there is a significant difference between execution time on the webserver and the system running in command line on e.g. a Pentium 4 machine on Red Hat 9.0. The execution time on the current webserver of CoolNELLI heavily depends on the workload at that specific moment.

Examples of processing times on the Galapagos Islands text 7 are shown in tables 6.3 and 6.4. The total time is calculated as follows:

$$\begin{aligned} \text{Total time} \quad = \quad & \text{NER-runtime} + \\ & max(\text{IRGoo-runtime}, \text{IRWiki-runtime}) + \\ & \text{Link-insertion-runtime} \end{aligned}$$

This is because the two IR modules are run in parallel.

From these processing times it is clear that the command line CoolNELLI runs a lot faster than through the web interface. The system is also suitable for processing incoming (or outgoing) e-mail. Please keep in mind that these running times are generated with a specific text example and therefore not completely representative for texts in general. Other issues like the nondeterministic character of the Internet (e.g. processing times for different queries in Google and Wikipedia) also influence the execution time of the system. The processing time in command line mode gives an indication that execution times will decrease significantly on a fast reliable dedicated webserver.

Module	Seconds
NER	11.35
IRGoo	26.90
IRWiki	35.97
Link insertion	00.24
Total	47.56

Table 6.3: Execution times in seconds of the different modules on the current webserver

Module	Seconds
NER:	1.48
IRGoo	6.07
IRWiki	6.51
Link insertion	0.11
Total	8.10

Table 6.4: Execution times in seconds of the different modules when run from the command line

Chapter 7

Appendix A: Test set 1

These are the texts we selected to evaluate the performance of CoolNELLI. They consist of two news articles, an informative text and an e-mail.

7.1 TEXT I

Justin Timberlake, Isaac Hayes To Join In Elvis Celebration
The day after Independence Day is when the fireworks will really be popping in Memphis, Tennessee.

As part of the city's festivities surrounding the 50th anniversary of the birth of rock and roll, Justin Timberlake, funk pioneer Isaac Hayes, Elvis guitarist Scotty Moore, actor/singer Billy Bob Thornton and others will gather at the legendary Sun Studios on July 5, 50 years to the day after Elvis Presley's "That's All Right" was recorded there in 1954. The bluesy, up-tempo song which featured Moore on guitar and Bill Black on bass, is widely considered to have started the rock revolution.

Timberlake, who was born in Memphis, has often credited Elvis as one of his main influences. At noon on the 5th, he, Moore, Hayes and "Wooly Bully" author Sam the Sham will take part in the "Global Moment in Time," during which radio stations across the world have agreed to play "That's All Right." Organizers hope they will set a record for the most simultaneous airplay for a single song.

The Memphis Convention & Visitors Bureau will host a daylong event that will feature performances and speeches. "We are thrilled that Justin is coming home to join us as we celebrate this musical milestone," Bureau president Kevin Kane said. "He is already a world-renowned artist, and his work continues to extend Memphis' musical influence to the next generation."

7.2 TEXT II

U.S. rallies allies against terror
President Bush yesterday said terrorists will "never shake the will of the United States," and the prime minister of the Netherlands said his country stands "shoulder to shoulder" with the United States to fight global terrorism.

In the wake of Spain's planned withdrawal of troops in Iraq after a suspected al Qaeda attack in Madrid last week, Mr. Bush used the Dutch leader's visit as an opportunity to rally the rest of his global allies in the war on terrorism. "Terrorists will kill

innocent life in order to try to get the world to cower," Mr. Bush said. "That's what they want to do. And they'll never shake the will of the United States. We understand the stakes, and we will work with our friends to bring justice to the terrorists."

Mr. Bush pointed out that al Qaeda not only has attacked Spain in the past year, but also Turkey and Saudi Arabia. "They kill wherever they can, and it's essential that the free world remain strong and resolute and determined," Mr. Bush said. Dutch Prime Minister Jan Peter Balkenende, speaking at a joint press conference with the president, said his country remains a staunch U.S. ally and urged other countries to continue to fight against terrorism. "It's important that the world society, the international community, stands shoulder to shoulder and shows its solidarity to fight against these terrible attacks," Mr. Balkenende said. "We share that same view, and we will work together."

The prime minister said he and Mr. Bush did not discuss what to do in Iraq after the coalition hands over sovereignty to a new Iraqi government on June 30, when Spain said it will pull its forces. Any decision about Dutch participation after that date, he said, would be made then. "That is the responsibility of the Dutch government and Dutch parliament, and we'll talk about it," Mr. Balkenende said. Almost all of the other governments helping to rebuild Iraq said they would stay the course. Polish Prime Minister Leszek Miller said abandoning Iraq now "would amount to an admission that the terrorists are right and that they are stronger than the civilized world." Leaders of Australia, Japan, Britain, Ukraine and Bulgaria also said they would not pull their troops out of Iraq. However, the defense secretary of Honduras said his country will withdraw the 370 troops that it had committed to aid the Spanish contingent in Iraq. Officials in El Salvador and Nicaragua, which also sent forces to supplement Spain's force of 1,300, said they'd remain in Iraq. Mr. Bush said he was confident the Netherlands, an influential European nation, would continue its staunch support.

7.3 TEXT III

The Galapagos Islands

The Galapagos Islands are an archipelago of some 13 volcanic islands and associated islets and rocks located in the Pacific Ocean about 1,000 kilometers west of the coast of South America. The Galapagos archipelago is politically part of Ecuador. The oldest of the islands are about 4 million years old and the youngest are still in the process of being formed. Indeed, the Galapagos islands are considered to be one of the most active volcanic areas in the world.

They are famed for their vast number of endemic species and the studies conducted by Charles Darwin that led to his theory of natural selection.

The islands are distributed to the north and south of the equator. The equator crosses the northern part of the largest island, Isabela.

The Galapagos Islands were declared a national park in 1959, protecting 97.5area was set aside for the four human settlements that existed at that time. Approximately 1,000 to 2,000 people called the islands their home. In 1972 a census was done in the Archipelago and a population of 3,488 was recorded. By the 1980s, this number had dramatically risen to more than 15,000 people.

In 1986 the surrounding ocean was declared a marine reserve. UNESCO recognised the islands as a World Heritage Site in 1978, which was extended in December 2001 to include the marine reserve. The Charles Darwin Foundation dedicated to the conservation of the islands was founded in Belgium in 1959. Noteworthy species include:

Land iguana, Conolophus subcristatus
Marine iguana, Amblyrhynchus cristatus (only iguana feeding from the sea)
13 endemic species of finch, popularly called Darwin's finches.

7.4 TEXT IV

Dear Ken,
As you know our first meeting will be on Tuesday, May 23 at 4:30 p.m.
Pete Peterson, chairman of the Blackstone Group, has kindly agreed to host it in his conference room. The address is:

The Blackstone Group
345 Park Avenue (between 51st and 52nd)
31st floor
New York, NY 10154
USA
(212)583-5000.

I know that some plans have changed since we first set the meeting, with some people being away and others having cancelled trips, so could you please reconfirm whether or not you will be able to attend? Sometime next week, we will be sending you an agenda and a background paper that focuses on the issues we propose to discuss. I greatly appreciate all the help members of the panel have given Tim Koller and me.

As you know, Tim Koller is the McKinsey&Co. partner who kindly prepared the draft outline we originally sent around. He will also be preparing the paper you will receive shortly. I am attaching the final list of panel members. Please do look it over. There are some significant additions: Kenneth Lay of Enron, Rob Glaser of RealNetworks, Len Baker of Sutter Hill Ventures, Dennis Powell of Cisco.

I look forward to your rsvp, and hopefully to seeing you on May 23.

Best regards,
Jeff

Chapter 8

Appendix B: Test set 2

This is another set of test texts to test CoolNELLI. It consists of a news article, an e-mail and an informative web page (no HTML is shown in this document).

8.1 TEXT I

Italy Says Will Return Ethiopian Obelisk in April
Thu Jan 27, 2005 11:54 AM ET
ROME (Reuters) - An Ethiopian national treasure, the ancient Axum Obelisk that was plundered by Italian fascist invaders in 1937, will be returned by Rome in April, Italy's Foreign Ministry said on Thursday.

The 24-meter obelisk, believed to be at least 1,700 years old, was split in three and hauled off when Italy under Fascist dictator Benito Mussolini invaded Ethiopia in 1937.

Italy promised in 1947 to return the 200-tongranite column, a symbol of the dawn of Ethiopian civilization, but arguments and logistical problems delayed it until November last year when the two countries finally agreed to fly it home.

"We expect the first flight with the first segment to happen in the first 10 days of March, the other two flights will follow around 20 days apart. In that way, the column will be returned by the end of April," an Italian Foreign Ministry spokesman said.

Ethiopia had previously said it expected the obelisk in May.

Returning the segments of the monument and the machinery to put it back together is a gargantuan logistical task.

Landlocked Ethiopia has had to build a special runway for the only aircraft big enough to carry the pieces, the U.S.-built C-5 Galaxy and Russian-made Antonov 124. The Antonov was the plane finally chosen to bring the obelisk home.

The column is a funerary monument from pre-Christian Ethiopia, the largest and finest from the ancient site of Axum in northern Ethiopia.

Ethiopia has said it plans to celebrate the obelisk's return with a national holiday.

8.2 TEXT II

Dear Hugo,

Our small company Royal Dutch is doing well and it is time to make expansion plans. It would be a good idea if we could meet and discuss these plans next Wednesday. I have invited both Henri Deterding as well as my brother Jean Baptiste to be present at this meeting. Among others I would like to discuss plans for oil drilling in Alaska. Explorations by Henri have shown that the Teshekpuk Lake area is a promising location for oil drilling. We should discuss transpostation costs and dealing with the local Inuits and Greenpeace.

Regards,
Jean

8.3 TEXT III

Robert De Niro From Wikipedia, the free encyclopedia.
Robert De NiroRobert De Niro (born August 17, 1943 in New York City) is an acclaimed American film actor who is noted for having starred in several of director Martin Scorsese's films. Praised for his commitment to his roles, De Niro gained 60 pounds (27 kg) and learned how to box for his portrayal of Jake LaMotta in Raging Bull, ground his teeth for Cape Fear, and learned to play the saxophone for New York, New York (all Scorsese films).

Although not articulate, De Niro is generally considered a skilled observer of physical tics and details, and an intense perfectionist.

A graduate of the Little Red School House, De Niro made his first film appearance in 1968 in Greetings (directed by Brian De Palma). After that, he played some major and minor roles in other films not widely seen, until he gained popularity with his role in Bang the Drum Slowly (1973). He began to work with Martin Scorsese in the same year when the two collaborated on Mean Streets. Later Scorsese films in which De Niro has participated are Taxi Driver (1976), New York, New York (1977), Raging Bull (1980), The King of Comedy (1983), Goodfellas (1990), Cape Fear (1991), and Casino (1995). In these films, De Niro has primarily played charming but emotionally unstable characters who have sociopathic tendencies.

In the mid-1980s, De Niro began expanding into occasional comedic roles, and has had much success in that area as well with such films as Brazil (1985), Midnight Run (1988), Wag the Dog (1997), Analyze This (1999), and Meet the Parents (2000).

He has won two Academy Awards: as Best Actor for his role in Raging Bull; and as Best Supporting Actor for The Godfather, Part II.

Interestingly, De Niro and Marlon Brando are the only pair of actors who have won Academy Awards for portraying the same person: De Niro won for playing young Vito Corleone in The Godfather, Part II, and Brando had won previously (although he declined the award) for playing the elderly Don Vito in The Godfather.

Contrary to popular belief De Niro is primarily Irish-American, not Italian. In October 2004 he canceled an appearance in Rome, Italy after Italian officials claimed he had presented negative stereotypes of their ethnicity in his films. In December De Niro attended an art exhibit in Rome where he presented his late father's art for the first time - Italian officials were surprised and claim that there are now no hard feelings.

De Niro is often compared to fellow iconic actor Al Pacino and they finally teamed together in Michael Mann's Heat (1995). De Niro played a younger version of Pacino's father in The Godfather, Part II.

In 2004 De Niro re-married his wife Grace Hightower.

Chapter 9

Appendix C: Communication between modules

The modules communicate using XML input and output. XML schemas provide a means for defining the syntax of XML documents. The schemas we used can be found in Section 9.1 to 9.4.

9.1 Control to NER

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Date: 2005/01/17 -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
xmlns="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
elementFormDefault="qualified">

<xs:complexType name="textType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="source" type="xs:anyURI" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="ltp">
    <xs:complexType>
      <xs:sequence>
<xs:element name="text" type="textType"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

9.2 NER to IR

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Date: 2005/01/27 -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
xmlns="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
elementFormDefault="qualified">

<xs:simpleType name="namedEntityTypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="properName"/>
      <!-- Infrequent noun means: infrequent in some test corpus -->
      <xs:enumeration value="infrequentNoun"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="namedEntitySubtypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="person"/>
      <xs:enumeration value="location"/>
      <xs:enumeration value="organisation"/>
      <xs:enumeration value="unknown"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="namedEntityType">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="type" type="namedEntityTypeType"/>
    <xs:element name="subtype" type="namedEntitySubtypeType" minOccurs="0"/>
    <xs:element name="title" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="offset" type="xs:positiveInteger" use="required"/>
  <xs:attribute name="length" type="xs:positiveInteger" use="required"/>
</xs:complexType>

<xs:complexType name="textType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="source" type="xs:anyURI" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="ltp">
    <xs:complexType>
      <xs:sequence>
```

```
<xs:element name="text" type="textType"/>
<xs:element name="namedEntity" type="namedEntityType" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

9.3 IRGoo to Control

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Date: 2005/01/27 -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
xmlns="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
elementFormDefault="qualified">

<xs:simpleType name="namedEntityTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="properName"/>
    <!-- Infrequent noun means: infrequent in some test corpus -->
    <xs:enumeration value="infrequentNoun"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="namedEntitySubtypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="person"/>
    <xs:enumeration value="location"/>
    <xs:enumeration value="organisation"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="confidenceURIType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="confidence" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="low"/>
              <xs:enumeration value="medium"/>
              <xs:enumeration value="high"/>
            </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
```

```
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="namedEntityType">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="type" type="namedEntityTypeType"/>
    <xs:element name="subtype" type="namedEntitySubtypeType" minOccurs="0"/>
    <xs:element name="title" type="xs:string" minOccurs="0"/>
    <xs:element name="gooURL" type="confidenceURIType" minOccurs="0"/>
    <xs:element name="googleHitcount" type="xs:positiveInteger"/>
  </xs:sequence>
  <xs:attribute name="offset" type="xs:positiveInteger" use="required"/>
  <xs:attribute name="length" type="xs:positiveInteger" use="required"/>
</xs:complexType>

<xs:element name="ltp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="namedEntity" type="namedEntityType" maxOccurs="unbounde
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

9.4 IRWiki to Control

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Date: 2005/01/27 -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
xmlns="http://lcg-www.uia.ac.be/twiki/bin/view/Main/WebHome"
elementFormDefault="qualified">

<xs:simpleType name="namedEntityTypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="properName"/>
    <!-- Infrequent noun means: infrequent in some test corpus -->
    <xs:enumeration value="infrequentNoun"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="namedEntitySubtypeType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="person"/>
    <xs:enumeration value="location"/>
```

```
    <xs:enumeration value="organisation"/>
    <xs:enumeration value="unknown"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="confidenceURIType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="confidence" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="low"/>
              <xs:enumeration value="medium"/>
              <xs:enumeration value="high"/>
            </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="namedEntityType">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="type" type="namedEntityTypeType"/>
    <xs:element name="subtype" type="namedEntitySubtypeType" minOccurs="0"/>
    <xs:element name="title" type="xs:string" minOccurs="0"/>
    <xs:element name="wikiURL" type="confidenceURIType" minOccurs="0"/>
    <xs:element name="description" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="offset" type="xs:positiveInteger" use="required"/>
  <xs:attribute name="length" type="xs:positiveInteger" use="required"/>
</xs:complexType>

<xs:element name="ltp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="namedEntity" type="namedEntityType" maxOccurs="unbounde
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Bibliography

[1] The Open Directory Project. *Open Directory Project* http://www.dmoz.org/

[2] Google Search Engine Optimization. *Internet Advertising homepage to increase traffic with Google by Jim Boykin* http://www.internet-advertising-marketing-manual.com/google-optimization.htm

[3] Google Information for Webmasters. *Google* http://www.google.com/webmasters/

[4] Google Web APIs. *Google* http://www.google.com/apis/

[5] Wikipedia, the Free Encyclopedia. *Wikipedia* http://en.wikipedia.org/

[6] Wiktionary, a collaborative project to produce a free multilingual dictionary in every language. *Wiktionary* http://en.wiktionary.org/