NTNU
Norwegian University of Science and Technology

Structured data extraction: separating content from noise on news websites

Mikel Arizaleta

Master of Science in Computer Science
Submission date: June 2009
Supervisor: Pinar Öztürk, IDI

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

Problem Description

Web search results are generally polluted by noisy data. Suppose you are interested in media coverage of fast food culture. You search for "news fast food", expecting to find news stories on fast food. However, among your top results is an advertisement site with a "news" section listing the latest offers from fast food chains. In general, a search result contains many irrelevant pages, where the search term is used in a different sense or does not occur in the core content of the page, but instead in a navigation link, an advertisement, an out-of-context user comment, etc.

One possible solution is to have specialized search engines for certain topics such as news, products, source code and so on. For this we need to be able to filter the noise from webpages and to extract the real content we are interested in. In this project we will work on news as a concrete example of this task. Our goal is to filter out the navigation links, advertisements, legal disclaimers and everything else except the title and text of the news item.

It is of course relatively easy to manually write code for extracting content from a particular webpage, or even from a whole domain like CNN News. However, this clearly does not scale and will quickly become infeasible when dealing with hundreds or thousands of news sites, all using their own format. Moreover, the code will break once a site decides to change its lay-out.

What we therefore want to do is automatically learn information extraction rules/patterns from a set of examples. This is called "automatic wrapper induction" when it uses supervised learning methods, that is, with examples correctly labeled by humans. When it is unsupervised (learns without human help) it is generally called "automatic data extraction". Structured data extraction is an active field of research and has many more applications besides facilitating web search. It combines methods from text processing, pattern matching and machine learning.

In this project we will first spent some time on the relevant literature to become familiar with the state-of-the-art. We will also have a look at some of the extraction software currently available, which we may want to reuse. Both, however, only as far as they are relevant for the purpose of mining news articles. We will start early with implementing a simple baseline system and creating a set of annotated examples for testing. Skills in scripting will definitely be an advantage here. We will then continue from this basis by analyzing errors and implanting new solutions, hopefully improving on the state-of-the-art results in the end. In terms of results, we are aiming for a functional and reusable implementation, and if possible a scientific publication.

Assignment given: 16. January 2009
Supervisor: Pinar Öztürk, IDI

Structured data extraction: separating content from noise on news websites

Mikel Arizaleta Delshorts

Contents

Chapter 1

Abstract

In this thesis, we have treated the problem of separating content from noise on news websites. We have approached this problem by using TiMBL, a memory-based learning software. We have studied the relevance of the similarity in the training data and the effect of data size in the performance of the extractions.

Chapter 2

Introduction

Web search results are generally polluted by noisy data. Suppose you are interested in media coverage of fast food culture. You search for "news fast food", expecting to find news stories on fast food. However, among your top results there is an advertisement site with a "news" section listing the latest offers from fast food chains. In general, a search result contains many irrelevant pages, where the search term is used in a different sense or does not occur in the core content of the page, but instead in a navigation link, an advertisement, an out-of-context user comment, etc.

One possible solution is to have specialized search engines for certain topics such as news, products, source code and so on. For this we need to be able to filter the noise from web pages and to extract the real content we are interested in. In this thesis we will work on news as a concrete example of this task. Our goal is to filter out the navigation links, advertisements, legal disclaimers and everything else except the title, the headings and text of the news item.

It is of course relatively easy to manually write code for extracting content from a particular web page, or even from a whole domain like CNN News. However, this clearly does not scale and will quickly become infeasible when dealing with hundreds or thousands of news sites, all using their own format. Moreover, the code will break once a site decides to change its lay-out.

What we therefore want to do is automatically learn information extraction rules/patterns from a set of examples. This is called *automatic wrapper induction* when it uses supervised learning methods, that is, with training examples correctly labeled by humans. When it is unsupervised (learning without human help) it is generally called "automatic data extraction". In this thesis we will explore the automatic wrapper induction approach. Structured data extraction in both approaches is an active field of research and has many more applications besides facilitating web search. It combines methods from text processing, pattern matching and machine learning.

The tool chosen to study from a practical point of view this problem is TiMBL, a software package for memory-based learning, and an associated suite of software tools for memory-based language processing.

In the development of this thesis, we divided the work in four parts:

i. Collect the raw data that will be used as training data for the learning system.

ii. Decide which is the best way to tag the data.

iii. Construct the training data in the format needed by TiMBL consisting of the selected features.

iv. Run the experiments and optimize the system.

In chapter 3 we will give some basic definitions in the data mining world, we will talk about automatic wrapper induction and machine learning and about some examples of it. Finally we will give some theoretical background on things (e.g., evaluation measures) we are using in the experimental chapter.

We will show how we collected the data and how we annotated them in chapter 4. In this chapter we will discuss two different types of annotation and why one is better than the other: Character-based annotation and Element-based annotation.

In chapter 5 we will explain what features we took and how we constructed the final training data merging the annotation schemes and the raw data.

Chapter 6 is the experiments chapter. We will divide the experiments in 3 types:

i. Extraction within single news domain: just training from the same source as the test file is used.

ii. Extraction merging domains: all training data are used in the predictions.

iii. Extraction across new domains: all training data except those from the same source as the test file is used.

We will study the TiMBL's performance with the default settings and we will look for the optimal ones. Finally we will study the effect of data size in the performance of the predictions.

To conclude this introductory chapter, I would like to thank some people who contributed directly or indirectly to this thesis. Erwin Marsi, my supervisor, for all his advices and guidance. He always was there when I needed anything. Pinar Öztürk, for all her administrative help. All my friends I met in Norway in this 6 incredible months. My family, for make possible this experience abroad and for their support during all my studies. Finally to my girlfriend, for putting up with me and for support me always.

Chapter 3

Background

3.1 Automatic wrapper induction

The Web is the single largest data repository ever available, providing access to numerous sources of useful information in textual form such as telephone directories, catalogs, news articles, etc. The text data in the Internet has increased massively and has become a chaos of millions of web pages. In spite of this chaos, it is possible to find some light at the end of the tunnel. Recently, there has been much interest in building systems that gather specific information on a user's behalf.

The set of processes needed to put some order into the data available nowadays and extract the high-quality information from it, is called text mining. In this thesis, we will focus on one type of text mining: automatic wrapper induction. Wrapper induction uses machine learning methods to generate extraction rules following the next three steps:

i. Given a set of texts (news articles web pages in this case), they are labeled by humans marking the target items. This labeled examples will be the training data for the learning system.

ii. The system learns extraction rules from this training data.

iii. The extraction rules are used by the system to automatically label unseen news articles.

Wrapper induction is used nowadays as a solution for many kind of problems. For examples in the news aggregator sites, where you can read a brief summary of the article together with the link to the full news. Due to the size of data they are using, it is impossible to do all the text labeling manually by humans. This aggregator sites use wrapper induction to collect every day all the articles published in the newspaper web sites and extract the most relevant information of them.

Another example is the price watching sites, like Kelkoo. This kind of sites collect data of thousands of products from the web sites that really sell the products and sort them by price, stock etc. Again, a wrapper induction system is behind this huge task of data collection.

Focusing on our problem, separating content from noise on news websites, the biggest advantage of using a learning system, is that it is possible to avoid the problem that represents the different formats and html structures of different web domain when it comes to create the extraction rules. Moreover, the structure and format of the web news domains is not stable over time, so it is not possible to write fixed extraction rules. The extraction rules learned by the system are not specific rules for a certain domain or for a certain html structure, but rules based on common features that belong to all the news articles. For example if we have a system based on fixed rules we are not able to extract the content of a page from a new domain. However, the news articles from this new domain will share lots of features with the already known articles, even if they are from different sources, so the learning system will be able to extract the content, of course, the accuracy of this extractions will depend on lots of factors, to study this dependence and the accuracy is one of the goals of this thesis.

Nowadays, there are already many approaches to wrapper induction, some of them are in the list below:

- WIEN (Kushmerick et al, 1997)
- Softmealy (Hsu and Dung, 1998)
- Stalker (Muslea et al. 1999)
- BWI (Freitag and McCallum, 2000)
- WL2 (Cohen et al. 2002)

- Thresher (Hogue and Karger, 2005)

We will explain briefly just the WIEN and Stalker.

WIEN was the first wrapper induction system (1997), it was implemented by The University of Washington. It works for both Web page and free text. The system defines 6 wrapper classes (templates) to express the structures of web sites, from all of them, the simplest and powerful one is LR (left-right) wrapper class. It uses left- and right- hand delimiter to extract the relevant information. Combining these 6 classes we can handle 70 % web sites.

The Stalker wrapper induction project (1999) was designed by The University of Southern California, this wrapper only works for Web pages. If we compare Stalker with WIEN, we can say that the first one is more expressive and efficient than WIEN. The system treats a web page as a tree-like structure and handles information extraction hierarchically. It uses disjunctions to deal with the variations, disjunctive rules are ordered list of individual disjuncts. The wrapper will successively apply each disjunct in the list until it finds one that matches.

3.2 Machine learning

Machine learning is a branch of the Artificial Intelligence whose objective is to develop techniques that enable machines to learn. The main goal of the machine learning is to create programs that are able to generalise behaviours using some structured information supplied as examples. It is thus a knowledge induction process.

The field where machine learning is applied is enormous, from new generation games with advanced AI integrated to weather forecast or traffic jams predictions.

3.2.1 Classification

The machine learning algorithms can be classified in two families, unsupervised and supervised learning.

The unsupervised algorithms try to determine how the data are organized, but to reach this goal it just use unlabeled examples. On the other hand, the supervised algorithms used training data (labeled) to create a learning function. The output of the function can be a continuous value (called regression), or can predict a class label of the input object (called classification). In the classification problems this function predicts the labels of an unseen object after having seen a number of examples.

We are going to study one family of the the supervised classification algorithm, the Memory-based learning (MBL).

3.2.2 Memory-based learning

Memory-based learning (MBL) is a machine learning method based on one of the most important learning strategies used by humans. In this way of learning, humans use experience to try to guess which one is the best choice confronted with a decision or problem. But humans do not extract rules from the experience and then apply these rules to solve the new problems, people use the experience directly, we look for the most similar experiences and we take the decision based on comparing the actual situation directly with the past ones. This is the approach of the MBL algorithms. Applying this ideas to a learning machine, MBL uses a simple storage system to keep the experience in memory and when a new decision has to be taken, the algorithm looks for the most common experiences and reuses the solution taken in these ones.

The other kind of learning system that extracts patterns or rules called e.g. Rule Induction, filters the exceptional, uncommon or low-frequency cases in the process to create general rules. This kind of learning methods are called *eager learning*. In many problems, *eager* approach is very useful, but in the problem of this thesis, the data extraction of news articles, we think that the atypical cases from the examples are very important for the decisions. For example, since we are interested in seeing what happens if we try to extract the content from a news article without having any article from the same source in the training data, we think that the more varied is the training data, the more accuracy we can get. This different point of view from the *eager learning* methods is called *lazy learning*. MBL is a lazy learning system which keeps all the data available for processing, even the exceptions.

Due to the *laziness* of MBL, if the data available are very big, the algorithm can be very slow, but MBL is acceptably fast for relatively small data-sets. Since for the experiments we are going to run, we do no expect to use more than 300 human-tagged news articles as training data, the time processing is not a problem.

An MBL system has two components, a *learning component* which is memory based and a *performance component* which is similarity based.

The *learning component* stores all the examples in memory, and as we said previously, MBL is a *lazy* learner, so it keeps all the examples in memory without abstraction or restructuring. The examples are stored in memory as a set of vectors containing all the features that describe the example and a label that classifies it.

In the performance component of an MBL system, the product of the learning component is used as a basis for mapping input to output. This usually takes the form of performing classification. During classification, a previously unseen test example is presented to the system. The similarity between the new instance X and all examples Y in memory is computed using some distance metric. The extrapolation is done by assigning the most frequent category within the found set of most similar example(s) (the k-nearest neighbors) as the category of the new test example.

To develop the goals of this thesis, we used a software package that implements a set of memory-based learning algorithms, the Tilburg Memory Based Learner (TiMBL from this point onwards).

TiMBL implements different metrics and algorithms. The most basic metric is Overlap Metric given on equations 3.1 and 3.1. The distance between two instances is simply the sum of the differences between the features.

$$\Delta(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{3.1}$$

where

$$\delta(x_i, y_i) = \begin{cases} abs(\frac{x_i - y_i}{max_i - min_i}) & \text{if numeric, else} \\ 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i \end{cases} \tag{3.2}$$

The Overlap metric counts the matching of the features vectors X and Y in the equations. If we don't know about the features relevance, this is a reasonable choice. Otherwise, we can use this knowledge changing the weight of the features, or looking at the behaviour of the features in the training data. We can compute statistics about the relevance of features by looking at which features are good predictors of the class labels.

Information Gain (IG) weighting looks at each feature in isolation, and measures how much information contributes to our knowledge of the correct class label. The Information Gain of feature i is measured by computing the difference in uncertainty between the situations without and with knowledge of the value of that feature 3.3.

$$w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v) \tag{3.3}$$

Where C is the set of class labels, V_i is the set of values for feature i , and $H(C) = -\sum_{c \in C} P(c) log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set.

You can also use the normalized version of Information Gain, called Gain Ratio, which is Information Gain divided by $si(i)$ (split info), the entropy of the feature-values, (Equation 3.5). This features weighting try to solve the main problem of Information Gain: the overestimation of the relevance of features with large numbers of values.

$$w_i = \frac{H(C) - \sum_{v \in V_i} P(v) \times H(C|v)}{si(i)} \tag{3.4}$$

$$si(i) = -\sum_{v \in V_i} P(v) log_2 P(v) \tag{3.5}$$

The resulting Gain Ratio values can then be used as weights w_i in the weighted distance metric (Equation 3.6).

$$\Delta(X,Y) = \sum_{i=1}^{n} w_i \delta(x_i, y_i) \tag{3.6}$$

TiMBL has two other ways to calculate the feature weighting: Chi-square and shared variance. The Chi-square is based on the χ^2 statistic. We can then either use the χ^2 values as feature weights in Equation 3.6, or we can explicitly correct for the degrees of freedom by using the Shared Variance measure (Equation 3.7).

$$SV_i = \frac{\chi_i^2}{N \times (min(|C|, |V_i|) - 1)} \tag{3.7}$$

Where $|C|$ and $|Vi|$ are the number of classes and the number of values of feature i, respectively, and N is the number of instances.

TiMBL implements a set of different algorithms. During the experiments we will explore the performance of two of them, IB1 and IGTREE. IB1 is the k-Nearest Neighbor (k-NN) algorithm with one of the different metric that we have explained before. The value of k-parameter represents the maximum distance that an example file can be from the test file to be used in the predictions. So of k-parameter is 1 (default) only the most common examples will be used. During chapter 6 we will study how k-parameter affects the performance of the predictions using the IB1 algorithm.

The other algorithm that we will study experimentally is IGTREE. This is a decision tree induction algorithm where the instance memory is restructured in such a way that it contains the same information as with the IB1, but in a compressed decision tree structure. The advantages of IGTREE over IB1 is that it uses less memory and it is faster, while IB1 usually gets better performance. In the experiment chapter (6) we will study the performance of both algorithms in this particular problem.

3.2.3 Evaluation measures

TiMBL provides a lot of output information about the problem, one could think that the most relevant one for the goals of this thesis is the accuracy, the overall percentage of correctly classified test instances, but as we will explain, in some problems accuracy is not a meaningful evaluation measure.

Accuracy per class label can be described as the number of times TiMBL has predicted the same class label as the one present in the test set. However, this measure presents a problem that we will explain by example: Imagine a problem with 2 classes to be predicted, A and B, and imagine that in the training data the distribution of both classes is 98 % A and 2 % B. This means that the distribution of class is very skewed. Always predicting class A you would get an accuracy rate of 98 % while you can not say that this is a good prediction system. As we will show in section 4.2, the problem we are studying presents this skeweed class distribution.

To guide us during the experiments and to know the real performance of the experiments we used other evaluation measures instead of accuracy: precision, recall and F-score. To describe this measures we need first to introduce some basic counts used in the advanced performance metrics.

- True positive (TP): count of examples that have a class C and are correctly predicted to have this class.

- False positive (FP): count of examples of a different class that the classifier incorrectly classified as C.

- False negative (FN): count of examples of class C for which the classifier predicted a different class.

- True negative (TN): count of examples with different classes that the classifier assigned a different class label than C.

With this previous basic metrics we can describe the following evaluations measures.

- $precision = \frac{TP}{TP+FP}$, or the proportional number of times the classifier has predicted correctly that some instance has class C.

- $recall = \frac{TP}{TP+FN}$, or the proportional number of times an example with class C in the test data has been classified as class C by the classifier.

- $F - score = \frac{2 \times precision \times recall}{precision + recall}$, or the harmonic mean of precision and recall.

The main evaluation measure we used to know if the predictions were being correct was the F-score, and during the experiment chapter we will focus on trying to optimize it.

TiMBL has another very useful functionality for our experiments. You can run a prediction test with the option cross-validation enable. This option lets you enter a list with a set of instance files, and TiMBL runs a test for each file using as training data all the others from the list. The output results file contain all the data from each test, so then you can calculate bias script global measures. We will talk about how we calculated this global measures in section 6.1.

Chapter 4

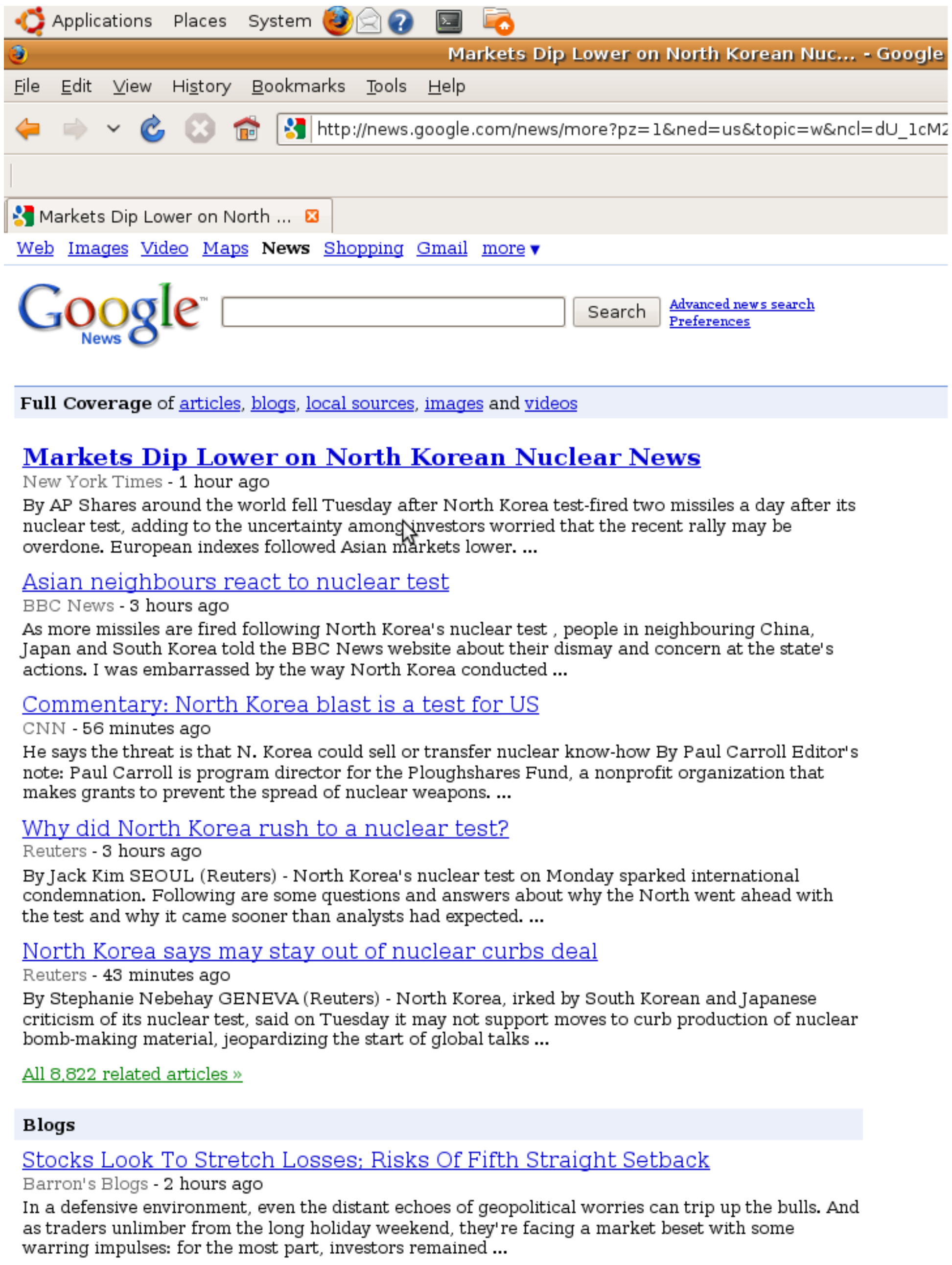
Data collection

4.1 Data source

The first problem we faced was to decide what type of data we wanted and how to get them. We needed some way to get lots of news text form news websites. In the Internet, there are thousands of news web pages, we decided to select the news from 16 representative news providers, all of them in English.

Since we need lots of news articles for the training data, we can not get them one by one from their own web page. To make our task easier, we decided to use a news agregator, where we could find the links for tons of news articles. The chosen one was Google News. The articles in Google News are sorted firstly by section: international, sports, business etc, and inside each section by cluster, there are 16 clusters with hundreds or thousands of articles each one. Inside each cluster the articles are shown as shown in figure 4.1 Each title is a link to the web page where the news article is, so a script was written to follow the *"all 8,822 related articles"* [1] link and get all the links of the cluster and select those from the 16 sources we work with.

[1] Just as a matter of interest, since we were automatically downloading hundreds of pages from Google domain in a very fast way, we were banned for 3 hours as a prevention of hacker attack. The problem was that the whole the lab where we were working was banned and some colleagues had to take a forced break.


Applications Places System
Markets Dip Lower on North Korean Nuc... - Google
File Edit View History Bookmarks Tools Help
http://news.google.com/news/more?pz=1&ned=us&topic=w&ncl=dU_1cM2
Markets Dip Lower on North ...
Web Images Video Maps News Shopping Gmail more
Google News
Search
Advanced news search
Preferences

Full Coverage of articles, blogs, local sources, images and videos

Markets Dip Lower on North Korean Nuclear News
New York Times - 1 hour ago
By AP Shares around the world fell Tuesday after North Korea test-fired two missiles a day after its nuclear test, adding to the uncertainty among investors worried that the recent rally may be overdone. European indexes followed Asian markets lower. ...

Asian neighbours react to nuclear test
BBC News - 3 hours ago
As more missiles are fired following North Korea's nuclear test , people in neighbouring China, Japan and South Korea told the BBC News website about their dismay and concern at the state's actions. I was embarrassed by the way North Korea conducted ...

Commentary: North Korea blast is a test for US
CNN - 56 minutes ago
He says the threat is that N. Korea could sell or transfer nuclear know-how By Paul Carroll Editor's note: Paul Carroll is program director for the Ploughshares Fund, a nonprofit organization that makes grants to prevent the spread of nuclear weapons. ...

Why did North Korea rush to a nuclear test?
Reuters - 3 hours ago
By Jack Kim SEOUL (Reuters) - North Korea's nuclear test on Monday sparked international condemnation. Following are some questions and answers about why the North went ahead with the test and why it came sooner than analysts had expected. ...

North Korea says may stay out of nuclear curbs deal
Reuters - 43 minutes ago
By Stephanie Nebehay GENEVA (Reuters) - North Korea, irked by South Korean and Japanese criticism of its nuclear test, said on Tuesday it may not support moves to curb production of nuclear bomb-making material, jeopardizing the start of global talks ...

All 8,822 related articles »

Blogs

Stocks Look To Stretch Losses; Risks Of Fifth Straight Setback
Barron's Blogs - 2 hours ago
In a defensive environment, even the distant echoes of geopolitical worries can trip up the bulls. And as traders unlimber from the long holiday weekend, they're facing a market beset with some warring impulses: for the most part, investors remained ...

Figure 4.1: Google News agregator

4.1.1 Annotation scheme

We can divide a news article in several parts, but of course, the more we divide it, the harder the classification will be. So we looked for a balance between good performance and usefulness. We decided to classify every single part of the web page into four classes: title, heading, body text and garbage. Distinguishing garbage from content is good for applications like search, where you do not want to index the noise. And distinguishing titles and heading is good for applications like multi-document summarization, where you do not want titles and headings to end up in the summary (Hahn, U. and Mani, I.[2000] The challenges of automatic summarization, 29–36)

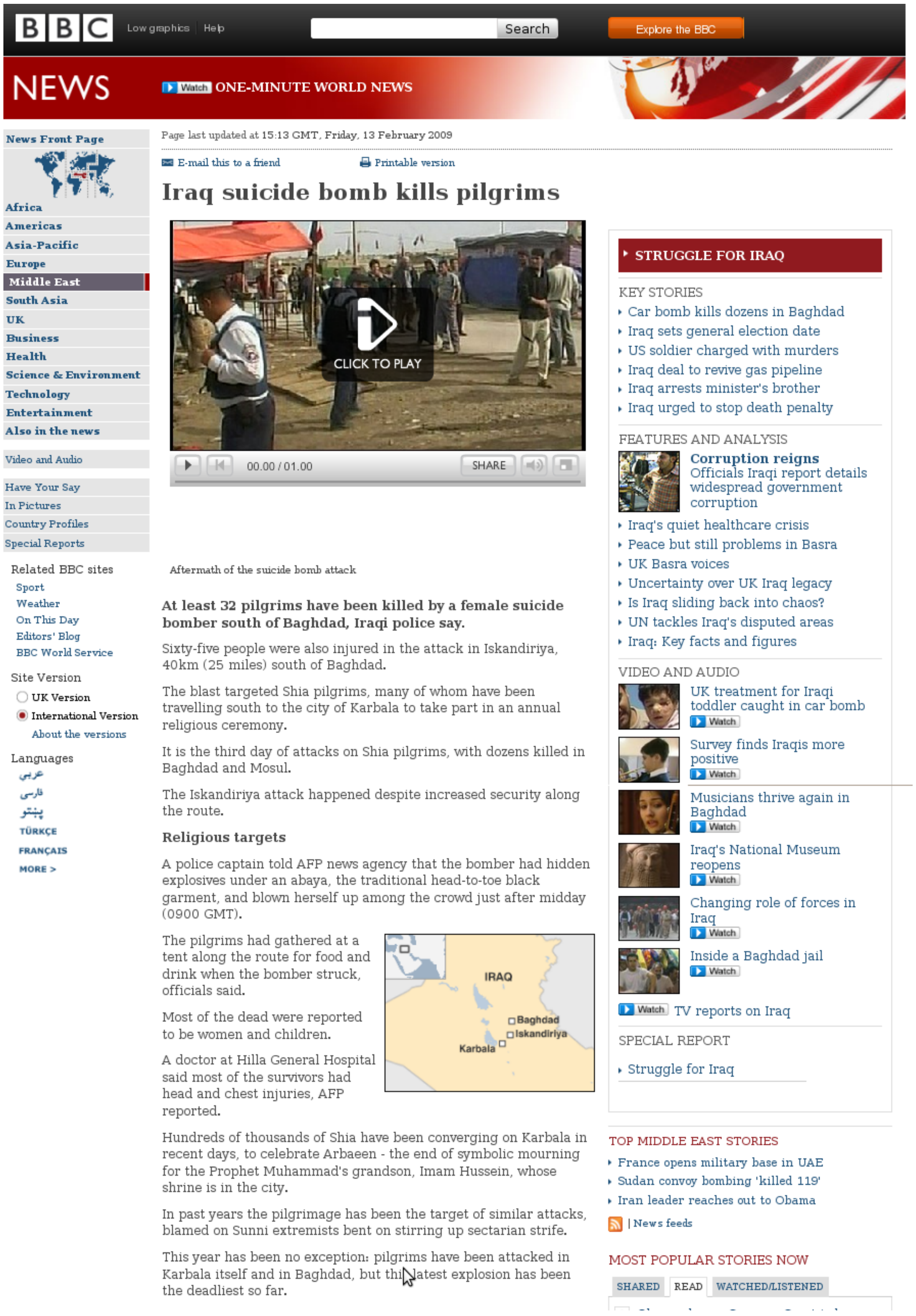
This basic classification let us realize these two important goals of the project.

To exemplify the annotation process we will use an example, in picture 4.2 (next page) you can see a typical article from a news web page, it comes from BBC.

Everybody should agree that the title of the news article is **Iraq suicide bomb kill pilgrims** , all the pictures, frames, advertisement, videos, banners, menus etc are garbage.

This could be easy to predict by a normal software based just on the tags of the html elements, provided that the structure is consistent, but as is well known, and as we have experienced during our own work, the structure and format of the articles from every source change quite often. The difficult part (and interesting) is the automatic classification of the remaining elements of the web. In this case, it is obvious that the comment of the video is not part of the news article, nor of the "KEY STORIES", "FEATURES AND ANALYSIS" and "TOP MIDDLE EAST STORIES" boxes.

Also, we should agree that **Religious targets** is a heading. However, even humans could disagree about the paragraph in bolt **At least 32 pilgrims have been killed ... police say.** It could be a heading or part of the body. we have regarded these kind of paragraphs as body. In the experiments chapter, we will see that the most difficult part of an article to be classified is the heading, and this web page is a good example of why. Finally, the remaining text is the body of the news article.



News Front Page
Africa
Americas
Asia-Pacific
Europe
Middle East
South Asia
UK
Business
Health
Science & Environment
Technology
Entertainment
Also in the news

Video and Audio

Have Your Say
In Pictures
Country Profiles
Special Reports

Related BBC sites
Sport
Weather
On This Day
Editors' Blog
BBC World Service

Site Version
UK Version
[✓ International Version]
About the versions

Languages
عربي
فارسی
پښتو
TÜRKÇE
FRANÇAIS
MORE >

Page last updated at 15:13 GMT, Friday, 13 February 2009

E-mail this to a friend   Printable version

Iraq suicide bomb kills pilgrims

Aftermath of the suicide bomb attack

At least 32 pilgrims have been killed by a female suicide bomber south of Baghdad, Iraqi police say.

Sixty-five people were also injured in the attack in Iskandiriya, 40km (25 miles) south of Baghdad.

The blast targeted Shia pilgrims, many of whom have been travelling south to the city of Karbala to take part in an annual religious ceremony.

It is the third day of attacks on Shia pilgrims, with dozens killed in Baghdad and Mosul.

The Iskandiriya attack happened despite increased security along the route.

Religious targets

A police captain told AFP news agency that the bomber had hidden explosives under an abaya, the traditional head-to-toe black garment, and blown herself up among the crowd just after midday (0900 GMT).

The pilgrims had gathered at a tent along the route for food and drink when the bomber struck, officials said.

Most of the dead were reported to be women and children.

A doctor at Hilla General Hospital said most of the survivors had head and chest injuries, AFP reported.

Hundreds of thousands of Shia have been converging on Karbala in recent days, to celebrate Arbaeen - the end of symbolic mourning for the Prophet Muhammad's grandson, Imam Hussein, whose shrine is in the city.

In past years the pilgrimage has been the target of similar attacks, blamed on Sunni extremists bent on stirring up sectarian strife.

This year has been no exception: pilgrims have been attacked in Karbala itself and in Baghdad, but this latest explosion has been the deadliest so far.

STRUGGLE FOR IRAQ

KEY STORIES
- Car bomb kills dozens in Baghdad
- Iraq sets general election date
- US soldier charged with murders
- Iraq deal to revive gas pipeline
- Iraq arrests minister's brother
- Iraq urged to stop death penalty

FEATURES AND ANALYSIS

Corruption reigns
Officials Iraqi report details widespread government corruption

- Iraq's quiet healthcare crisis
- Peace but still problems in Basra
- UK Basra voices
- Uncertainty over UK Iraq legacy
- Is Iraq sliding back into chaos?
- UN tackles Iraq's disputed areas
- Iraq: Key facts and figures

VIDEO AND AUDIO
- UK treatment for Iraqi toddler caught in car bomb — Watch
- Survey finds Iraqis more positive — Watch
- Musicians thrive again in Baghdad — Watch
- Iraq's National Museum reopens — Watch
- Changing role of forces in Iraq — Watch
- Inside a Baghdad jail — Watch

Watch TV reports on Iraq

SPECIAL REPORT
- Struggle for Iraq

TOP MIDDLE EAST STORIES
- France opens military base in UAE
- Sudan convoy bombing 'killed 119'
- Iran leader reaches out to Obama

| News feeds

MOST POPULAR STORIES NOW

SHARED READ WATCHED/LISTENED

Figure 4.2: News article from BBC

Distinguishing these text elements is a very easy task for humans, but quite hard for computers, who have no idea about the meaning or the visual layout. Computers can only read the html source of the news, so a way to classify the html soup is needed. Here you can see a piece of the html source of the same page:

```
<p>This year has been no exception: pilgrims have been attacked in Karbala itself
    and in Baghdad, but this latest explosion has been the deadliest so far.
<p><b>Female militants</b>
<p>It was a grim reminder that despite the considerable general improvement in
    security in Iraq there are still people out there bent on igniting sectarian
    passions - something many Iraqis had hoped was becoming a thing of the past,
    says the BBC's Jim Muir in Baghdad.
<p>Iraqi militants have increasingly used women to carry out suicide attacks as
    they are less likely to be searched than men.
<p>In 2007, there were eight suicide attacks by women; in 2008 there were 32, the
     US military says. In early January, a female bomber killed at least 35
    Shia pilgrims in a blast near a Baghdad shrine.
<p>Iraqi officials arrested an alleged militant recruiter last month.
<p>Samira Jassim allegedly recruited more than 28 women to blow themselves up in
    various parts of Iraq.
<p>
<!-- E BO -->
<br /><br clear="all" />
                              <div id="socialBookMarks" class="sharesb">
<h3>Bookmark with:</h3>
<ul>
<li class="delicious">
<a id="delicious" title="Post this story to Delicious" href="http://del.icio.us/post?
  url=http://news.bbc.co.uk/2/hi/middle_east/7887881.stm&amp;title=Iraq suicide
    bomb kills pilgrims">Delicious</a>
</li>
<li class="digg">
<a id="digg" title="Post this story to Digg" href="http://digg.com/submit?url=http:
    //news.bbc.co.uk/2/hi/middle_east/7887881.stm&amp;title=Iraq suicide bomb
    kills pilgrims">Digg</a>
</li>
<li class="reddit">
<a id="reddit" title="Post this story to reddit" href="http://reddit.com/submit?url=
    http://news.bbc.co.uk/2/hi/middle_east/7887881.stm&amp;title=Iraq suicide bomb
    kills pilgrims">reddit</a>
</li>
```

Now, the task of classifying the content of the web page into title, heading, body and garbage becomes harder. You have to find into the html soup the components that were easy to find in the visual layout and tag them in a machine readable way.

The annotation has to fulfill these three requirements:

i. Given the annotation of a web page, it should be possible to extract the text from the web page in a fully automatic fashion, without only human intervention.

ii. The annotation should allow us to regard the problem of text extraction as a classification problem, which can be learned using general machine learning techniques.

iii. Since machine learning requires quite a lot of annotated data, the annotation must be convenient to humans.
If it's true that is not a big problem from the results point of view, is very important from the time consuming point of view.

4.1.2 Character-based annotation

The first idea to annotate the data was to interpret the html document as a one long string, and create another file with annotations about where the title, body and headings are. In this file you had the exact position of each part of the news, i.e. the position of the first and last character inside the string. Following the same example of the BBC web page, the annotation would be like scheme 4.1 shows.

Table 4.1: Character-based annotation

title:	2151	2183
body1:	2512	2782
body2:	2986	3154
heading1:	3307	3323
body3:	3400	3587
...		

Some macros in Lisp were written (the e-macs macro's language), with them you just had to highlight in the html source the different parts of the news and they created the annotation file.

The advantage of this character-based annotation is that since you are selecting character by character, you can distinguish perfectly the part of the news from the garbage while with the element-base annotation sometimes it is impossible such as we will explain later.

This kind of annotation had two problems, the first one was that interpreting the whole document as a string, the structure of the html file was being skipped, and it seemed that this was going to be one of the strongest features. The second problem was that finding all the parts of the news in a html source was more difficult than expected, and with this method you should have to highlight all the sentences of the news one by one, i.e. a time-consuming problem.

4.1.3 Element-based annotation

This way of tagging is based in the html structure of the web page. The Python type elementtree is a container object designed to store hierarchical data structures in memory. You can parse an html document and create a tree where each node is an element of the html source.

Every element in html language has two parts where text can be written, they are called text and tail, so, all the letters we can see in a web page are written in one text or tail of one element. Since we are classifying the news in 4 pats, title(t), heading(h), body(b) and garbage(g), each element can be tagged with one of the possible combinations of them (one for the text and one for the tail), t+t, t+h, t+b, t+g, h+t, h+h, h+b, h+g, etc... 16 possible tags in total.

It could seem a very hard task tag one by one all the nodes of the html tree but since the majority of them were in the type of g+g (see section 4.2 for more details) and you can know that some kind of nodes will never contain a part of the news, it was not so hard.

Following this idea an script was written to take the html file and break it into nodes by text/tail. This file is the human annotation file (.hra) that contains all the text and tails of all the nodes (skipping that ones that can not contain the news such as *script, style ...*) and tagged as g+g, so you just have to change manually the nodes that contain part of the news, it means changing some dozens of tags per file.

The scheme 4.3 is a piece of an .hra file from the same news shown before.

The only problem this way has is that you have to tag the whole text or tail, but part of it can be body and another part can be garbage. However, during the experiments we saw that this case is very rare: it is very uncom-

```
This year has been no exception: pilgrims have been attacked in Karbala itself and in
Baghdad, but this latest explosion has been the deadliest so far.
#285 <p> text b
--------------------------------------------------------------------------------
Female militants
#287 <b> text h
--------------------------------------------------------------------------------
It was a grim reminder that despite the considerable general improvement in security in
Iraq there are still people out there bent on igniting sectarian passions - something many
Iraqis had hoped was becoming a thing of the past, says the BBC's Jim Muir in Baghdad.
#288 <p> text b
--------------------------------------------------------------------------------
Iraqi militants have increasingly used women to carry out suicide attacks as they are less
likely to be searched than men.
#289 <p> text b
--------------------------------------------------------------------------------
In 2007, there were eight suicide attacks by women; in 2008 there were 32, the US military
says. In early January, a female bomber killed at least 35 Shia pilgrims in a blast near a
Baghdad shrine.
#290 <p> text b
--------------------------------------------------------------------------------
Iraqi officials arrested an alleged militant recruiter last month.
#291 <p> text b
--------------------------------------------------------------------------------
Samira Jassim allegedly recruited more than 28 women to blow themselves up in various parts
of Iraq.
#292 <p> text b
--------------------------------------------------------------------------------
Bookmark with:
#298 <h3> text g
--------------------------------------------------------------------------------
Delicious
#301 <a> text g
--------------------------------------------------------------------------------
Digg
#303 <a> text g
--------------------------------------------------------------------------------
reddit
#305 <a> text g
--------------------------------------------------------------------------------
Facebook
#307 <a> text g
--------------------------------------------------------------------------------
StumbleUpon
#309 <a> text g
```

Figure 4.3: Human annotation file (hra)

mon to find a text or a tail with some garbage and some part of the news. We fount just a few cases where the last paragraph of the article contains the name of the author and both, the body and the name, are in the same text/tail.

As you can see the element-based annotation one was much better than the character-based one. Now, with the hra file and the html file we had all the information of the news, all the format, the structure and the manual tagging, so we needed to decided which features were going to be used and create the instance files (.inst) in the format that TiMBL needs, this will be explained in chapter 5.

4.2 Data properties

Finally, 375 pages were collected, 222 of them were annotated following the procedure explained before. In all this annotation files, the distribution of the tags in the text/tail of the elements was as follows in table 4.2:

Table 4.2: Tag distribution

tag	number	%
g	40472	91,04 %
t	221	0,49 %
h	126	0,28 %
b	3632	8,17 %

It means that almost all the content of the html source from a news web page is not part of the news. The previous table is interesting, but for us, it is more important the distribution of the "double" tags, because as we said, we are going to work with them. In table 4.3 you can see this distribution. The first thing we notice is that there are 9 tags never used, g+t, t+t, t+h, t+b, h+t, h+h, h+b, b+t and b+h. So since now, they are not going to be shown in the results.

Table 4.3: Double tag distribution (class)

tag	number	%
g+g	134782	97.318
g+t	0	0.000
g+h	2	0.001
g+b	624	0.451
t+g	221	0.160
t+t	0	0.000
t+h	0	0.000
t+b	0	0.000
h+g	114	0.082
h+t	0	0.000
h+h	0	0.000
h+b	0	0.000
b+g	2535	1.830
b+t	0	0.000
b+h	0	0.000
b+b	218	0.157

The important conclusion of double tag distribution is that the high rate of garbage tag among the data makes the TiMBL accuracy results not meaningful, because just tagging all the elements with g+g you could get a 97.318 % of accuracy, while any part of the news would be tagged correctly. In order to fix this problem a script that processes the results correctly was written, to read more about the TiMBL output and the results process see section 6.1.

Chapter 5

Feature construction

Using TiMBL, the choice of the features is highly relevant, because it tells TiMBL what it should "learn" and use to make the predictions, the performance of this predictions depends very much on this choice.

The first feature thought was the html structure, this is not a human-sensitive feature, i.e., reading real news in an online newspaper, you can not know what kind of html tags are behind each part of the news. Actually, it doesn't seem to be a strong feature to distinguish between the text part of the web page, because all of them can have the same tag structure. However, this is a very strong feature to avoid the advertising, pictures, and all non text elements of the web page since the tags are totally different.

Of course, it's impossible to get all the html tag structure, but we can add the tags of the neighbourhood elements of the current element. We decided to take an initial window of 5 tags before and 5 tags after, plus the current tag, i.e. 11 features. For each element of the html file, these 11 features represent the neighbours structure.

After the structure features were selected, we started to think of how humans recognize the body, title and headings of an online news. For us it is quite easy to separate the garbage from the real news just having a look. So the question is, which are these features we noticed so easily?

The most important ones seem to be the lay out in the web page, the length of the text and the format of the text. So we wondered which one of them we can add as a feature to TiMBL. The position is quite complicated, because it's difficult to know from the html source where an element is going to be placed. Furthermore, humans use this feature to avoid all the images, advertising etc.. and these things are correctly tagged by TiMBL using the html structure features.

The length one is very easy to get and it seems to be really strong, even more if we think that most of the text and tail parts of the garbage elements have a very short text or no text. This time, a window of 3 tails before and 3 tails after were added as a feature as well as 3 texts before and after, i.e. 14 length features.

The format of the text should be very important to separate the headings and the title from the body. The question was which features of the format should we take into account. When you take a look on a news, you identify the headings because they are in capital letters, the text is usually shorter than a normal paragraph and the sentence finishes, usually again, without a dot, a question mark, exclamation mark, quotation mark etc ...

So, this two new features were added, we considered that a text/tail is in upper letters if more than 70 % of the letters are, this time we decided not to add a window with the upper letter features because it is quite often to find some garbage text in upper letters.

The dot feature seems to be very strong, first of all to separate the headings from the body as we explained before, but it is also important to identify the body and separate it from the garbage, if you notice all the paragraph ends with one of the marks we commented before. Again a window of 14 dot features were added following the same idea of the length ones.

Once all the features where chosen, we began to create the instance files. In table 5.1 an example of one initial instance file with the first 11 features is shown.

Every element from the ElementTree is one instance, actually, the instance is the set of features that represents this element. Everyone of this set of features must be tagged as t+t,t+h,t+b,h+g... A script was written to extract the features from the .html file and tag every instance with the

Table 5.1: Piece of instance file with the first 11 features

					current element						**class**
#	#	#	p	b	**p**	p	p	#	p	p	**b+g**
#	#	p	b	p	**p**	p	#	p	p	b	**b+g**
#	p	b	p	p	**p**	#	p	p	b	p	**b+g**
p	b	p	p	p	**#**	p	p	b	p	#	**g+g**
b	p	p	p	#	**p**	p	b	p	#	dif	**b+g**
p	p	p	#	p	**p**	b	p	#	dif	tr	**g+g**
p	p	#	p	p	**b**	p	#	dif	tr	td	**h+g**
p	#	p	p	b	**p**	#	dif	tr	td	div	**b+g**
#	p	p	b	p	**#**	dif	tr	td	div	img	**g+g**
p	p	b	p	#	**dif**	tr	td	div	img	#	**g+g**
p	b	p	#	dif	**tr**	td	div	img	#	p	**g+g**

human annotation from the .hra file. This file was named .inst and contains all the data that TiMBL needs to do the predictions.

We decided to use the column format, i.e. every line is one instance and every column of the line is one feature. We had 41 features, so 41 columns, the last column is the tag of the instance. You can see this construction clearly in table 5.1.

Chapter 6

Experiments

6.1 Evaluation measures

Before starting with the experiments, a previous step was necessary, a processor for the result of TiMBL.

All experiments were carried out using cross validation and precision, recall and F-score. When using Timbl for testing, it can calculate a number of performance measures (using the +vas command line switch). These include the raw number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), as well as per class and overall precision, recall and F-score. However, when using TiMBL for cross validation, it can only produce these statistics per fold, not over all folds. Therefore, we had to write a script which reads the statistics per fold and calculates the corresponding overall statistics. We explain the details below.

- The precision, recall and F-score per news article element, i.e. per title, heading and body. This metric was necessary due to the tag distribution. As it was explained in section 4.2, 97.318 % of the instances are tagged with g+g by humans, so if we calculate a global measure of performance taking in consideration the g+g classes, the results would be much better than they really are, since just tagging all the instances with g+g we could have a global F-score of 97,318 %. However, with this new measure we realize that the real F-score would be 0 % in title, heading and body. So we calculate the TP, FP, TN and FN adding from all the classes related to a particular element of the article. For example, to calculate the performance of the title in the predictions

of BBC, we added the counts of the t+g, t+t, t+h and t+b classes from all the BBC news articles tested, and with them, we calculated the precision, recall and F-score for the title.
Another important reason to calculate this measures per title, heading and body insted of class, is that it better reflects the task of text extraction from webpages. That is, a class like b+g tags is just an intermediary representation that we need to turn the problem of text extraction into a classification problem, but predicting it is not a goal itself.

- The macro-mean precision, recall and F-score. This is a global metric, that tell us what performance is expected if we add a new domain. Each macro-mean is calculated by taking the average of the precision, recall and F-score respectively, over all the data tested.

- The micro-mean precision, recall and F-score. This is a global metric that shows the expected performance for a random instance. To calculate each micro-mean, you have to use the formulas shown in section 3.2.3 for precision, recall and F-score using global TP, FP, TN and FN calculated by adding all the particular ones.

- Another kind of result analysis was the error per file, a new file was created for each html file, this file (.error) contains all the errors in the predictions of TiMBL, i.e. when the tag predicted by TiMBL is different from the tag written by humans. These files were very useful when it comes to fix some human tag errors and to guide us during the feature construction

6.2 Extraction within a single news domain

As it is explained in section 3.2.2, the similarity between the training data that TiMBL is using and the test data is very important. The most similar is the data, the best performance you should get.

Following this idea, the first experiments were run using a number of pages from a certain news source as training data and predict the structure of unseen pages from the same news source. The experiments were run with the default configuration of TiMBL, i.e. with IB1 algorithm, feature weighting with Gain Ratio, weighted overlap and number of nearest neighbours (k-parameter) equal 1.

In the tables 6.1, 6.2 and 6.3 you can see all the results obtained in this first experiment.

Table 6.1: Results on title extraction within a single news domain

SOURCE	# FILES	TP	FP	TN	FN	prec	recall	fscore
aljazeera	18	18	0	8952	0	1.000	1.000	**1.000**
guardian	17	15	16	14360	2	0.484	0.882	**0.625**
BBC	23	22	1	14618	0	0.957	1.000	**0.978**
CNN	10	7	2	8403	2	0.778	0.778	**0.778**
xinhuanet	20	19	0	3723	1	1.000	0.950	**0.974**
latimes	10	10	2	5703	0	0.833	1.000	**0.909**
nytimes	15	15	1	9217	0	0.938	1.000	**0.968**
reuters	17	17	0	11640	0	1.000	1.000	**1.000**
theautralian	26	26	0	21721	0	1.000	1.000	**1.000**
timesonline	16	16	0	8302	0	1.000	1.000	**1.000**
dw-world	10	9	4	4390	1	0.692	0.900	**0.783**
fox	4	4	0	1992	0	1.000	1.000	**1.000**
hindu	5	5	0	1604	0	1.000	1.000	**1.000**
upi	11	11	0	7471	0	1.000	1.000	**1.000**
voamerica	12	12	0	7195	0	1.000	1.000	**1.000**
washintonpost	8	8	0	8589	0	1.000	1.000	**1.000**
total	222	214	26	137880	6			
macro-mean						0.918	0.969	**0.938**
micro-mean						0.892	0.973	**0.930**

The overall results were quite good, at least, better than we had expected. The results of Aljazeera were almost perfect, while in The Guardian, which was the worst one, title and heading predictions had plenty of errors.

These differences in the performance on the different sources are mainly due to the presence or lack of consistency in the formatting of the article. For instance, in Aljazeera, all the news have a similar layout and format, while there are other sources with different kind of articles. The differences in the performance don't seem to be related to the amount of training material, since some sources as Upi, Voice of America or The Washintong post have better performance than The Guardian or BBC. In this kind of experiments, the similarity between the training data is more relevant than the amount of it.

Table 6.2: Results on heading extraction within a single news domain

SOURCE	# FILES	TP	FP	TN	FN	prec	recall	fscore
aljazeera	18	27	0	8943	0	1.000	1.000	**1.000**
guardian	17	2	4	14384	3	0.333	0.400	**0.364**
BBC	23	42	9	14589	1	0.824	0.977	**0.894**
CNN	10	0	0	0	0	(nan)	(nan)	**(nan)**
xinhuanet	20	0	0	0	0	(nan)	(nan)	**(nan)**
latimes	10	0	0	0	0	(nan)	(nan)	**(nan)**
nytimes	15	0	0	0	0	(nan)	(nan)	**(nan)**
reuters	17	5	0	23307	2	1.000	0.714	**0.833**
theautralian	26	17	2	21727	1	0.895	0.944	**0.919**
timesonline	16	7	0	8311	0	1.000	1.000	**1.000**
dw-world	10	0	0	3735	0	(nan)	(nan)	**(nan)**
fox	4	0	0	0	0	(nan)	(nan)	**(nan)**
hindu	5	0	0	0	0	(nan)	(nan)	**(nan)**
upi	11	0	0	0	0	(nan)	(nan)	**(nan)**
voamerica	12	0	0	0	0	(nan)	(nan)	**(nan)**
washintonpost	8	5	0	8592	0	1.000	1.000	**1.000**
total	222	105	15	99853	7			
macro-mean						0.865	0.862	**0.859**
micro-mean						0.875	0.938	**0.905**

Table 6.3: Results on body extraction within a single news domain

SOURCE	# FILES	TP	FP	TN	FN	prec	recall	fscore
aljazeera	18	326	2	17608	4	0.994	0.988	**0.991**
guardian	17	252	10	28518	6	0.962	0.977	**0.969**
BBC	23	445	3	28813	21	0.993	0.955	**0.974**
CNN	10	191	9	24181	7	0.955	0.965	**0.960**
xinhuanet	20	148	5	7332	1	0.967	0.993	**0.980**
latimes	10	148	9	16983	5	0.943	0.967	**0.955**
nytimes	15	266	5	18193	2	0.982	0.993	**0.987**
reuters	17	168	3	23143	0	0.982	1.000	**0.991**
theautralian	26	645	7	64573	16	0.989	0.976	**0.982**
timesonline	16	238	4	24700	12	0.983	0.952	**0.967**
dw-world	10	16	0	4372	16	1.000	0.500	**0.667**
fox	4	31	1	1964	0	0.969	1.000	**0.984**
hindu	5	46	1	3166	5	0.979	0.902	**0.939**
upi	11	111	2	14848	3	0.982	0.974	**0.978**
voamerica	12	122	2	14286	4	0.984	0.968	**0.976**
washintonpost	8	108	2	17082	2	0.982	0.982	**0.982**
total	222	3261	65	309762	104			
macro-mean						0.978	0.943	**0.955**
micro-mean						0.980	0.969	**0.975**

Anyway, the body F-score was very good in all the sources. Actually, we had already thought that the body results should be better than title and heading. The main reason was the length feature, this should be a strong feature since it is the biggest difference between the text/tail from the body and the rest of text/tail. In section 6.5 you can see exactly the weights of all features and how much important the length one is.

6.3 Extraction merging domains

Once we had done the experiments using the training data from the same source as the test file, the next step in the experimentation was merge all the training data and run a cross-validation. It means, predict each file using all the rest from all the sources as training data (included the own source of the test file).

It could seem that the results of this experiments will be worst because inside the training data there is not just the "good" data, i.e. the data from the same source, but since they are run with k-parameter equal 1, it means that just the most similar training data are used to make the predictions, so at least, the results should be as good as the previous one.

In tables 6.4, 6.5 and 6.6 you can see the results for this experiment. In these tables we have omitted the TP, FP, TN, FN values but we have introduced the values of the different in the prediction, recall and F-score between the results of the extraction within single news domains and merging domains.

As we expected, the overall performance is better, even if in some tags of some sources there is a small decrease. For instance, the F-score of the title on BBC, heading in Aljazeera and body in CNN and reuters is a bit better extracting data within a single domain than merging domains, but a great majority of particular F-scores and macro-mean and micro-mean are better with merging domains.

If we look at the last rows of the tables 6.4, 6.5 and 6.6 we can see the global performance expressed by the F-score of the macro-mean and micro-

Table 6.4: Results on title extraction merging domains

SOURCE	# FILES	prec(Δ)	recall(Δ)	fscore(Δ)
aljazeera	18	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
guardian	17	1.000(+0.516)	0.882(+0.000)	**0.938(+0.313)**
BBC	23	0.955(-0.002)	0.955(-0.045)	**0.9(-0.078)**
CNN	10	0.875(+0.097)	0.778(+0.000)	**0.824(+0.046)**
xinhuanet	20	1.000(+0.000)	1.000(+0.050)	**1.000(+0.026)**
latimes	10	1.000(+0.167)	0.900(-0.100)	**0.947(+0.038)**
nytimes	15	1.000(+0.062)	1.000(+0.000)	**1.000(+0.032)**
reuters	17	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
theautralian	26	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
timesonline	16	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
dw-world	10	1.000(+0.308)	0.900(+0.000)	**0.947(+0.164)**
fox	4	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
hindu	5	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
upi	11	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
voamerica	12	1.000(+0.000)	0.917(-0.083)	**0.957(-0.043)**
washintonpost	8	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
macro-mean		0.989(+0.071)	0.958(-0.011)	**0.970(+0.032)**
micro-mean		0.991(+0.099)	0.964(-0.009)	**0.977(+0.047)**

Table 6.5: Results on heading extraction merging domains

SOURCE	# FILES	prec(Δ)	recall(Δ)	fscore(Δ)
aljazeera	18	1.000(+0.000)	0.963(-0.037)	**0.981(-0.019)**
guardian	17	0.400(+0.067)	0.400(+0.000)	**0.400(+0.036)**
BBC	23	0.929(+0.105)	0.907(-0.070)	**0.918(+0.024)**
CNN	10	0.000	(nan)	**(nan)**
xinhuanet	20	(nan)	(nan)	**(nan)**
latimes	10	(nan)	(nan)	**(nan)**
nytimes	15	(nan)	(nan)	**(nan)**
reuters	17	1.000(+0.000)	1.000(+0.286)	**1.000(+0.167)**
theautralian	26	0.900(+0.005)	1.000(+0.056)	**0.947(+0.028)**
timesonline	16	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
dw-world	10	1.000	0.750	**0.857**
fox	4	(nan)	(nan)	**(nan)**
hindu	5	(nan)	(nan)	**(nan)**
upi	11	0.000	(nan)	**(nan)**
voamerica	12	0.000	(nan)	**(nan)**
washintonpost	8	1.000(+0.000)	1.000(+0.000)	**1.000(+0.000)**
macro-mean		0.904(+0.039)	0.877(+0.015)	**0.888(+0.029)**
micro-mean		0.907(+0.032)	0.922(-0.016)	**0.915(+0.01)**

Table 6.6: Results on body extraction merging domains

SOURCE	# FILES	prec(Δ)	recall(Δ)	fscore(Δ)
aljazeera	18	0.994(+0.000)	0.985(-0.003)	**0.989(-0.002)**
guardian	17	0.973(+0.011)	0.981(+0.004)	**0.977(+0.008)**
BBC	23	0.993(+0.000)	0.983(+0.028)	**0.988(+0.014)**
CNN	10	0.950(-0.005)	0.955(-0.010)	**0.952(-0.008)**
xinhuanet	20	0.955(-0.012)	1.000(+0.007)	**0.977(-0.003)**
latimes	10	0.974(+0.031)	0.967(+0.000)	**0.970(+0.015)**
nytimes	15	0.996(+0.014)	0.985(-0.008)	**0.991(+0.004)**
reuters	17	0.988(+0.006)	1.000(+0.000)	**0.994(+0.003)**
theautralian	26	0.994(+0.005)	0.965(-0.011)	**0.979(-0.003)**
timesonline	16	0.980(-0.003)	0.972(+0.020)	**0.976(+0.009)**
dw-world	10	1.000(+0.000)	0.969(+0.469)	**0.984(+0.317)**
fox	4	0.969(+0.000)	1.000(+0.000)	**0.984(+0.000)**
hindu	5	1.000(+0.021)	0.980(+0.078)	**0.990(+0.051)**
upi	11	0.982(+0.000)	0.974(+0.000)	**0.978(+0.000)**
voamerica	12	0.992(+0.008)	0.976(+0.008)	**0.984(+0.008)**
washintonpost	8	0.982(+0.000)	1.000(+0.018)	**0.991(+0.009)**
macro-mean		0.983(+0.005)	0.981(+0.038)	**0.981(+0.026)**
micro-mean		0.985(+0.005)	0.978(+0.009)	**0.981(+0.006)**

mean, these are the most interesting results. Firstly because you can get a better picture of the real performance and secondly because these results are much more meaningful from a practical point of view than the ones from the section 6.2 since we are using all the training data we have in our hands.

Without optimizing the TiMBL settings, using the default ones we are in 97,7 % of F-score in the title, 91,5 % in the heading and 98,1 % in the body, these are quite promising results. Of the 3 parts of a news (title, heading and body), the most problematic one was being the heading, this is just what we expected because the variability in the format of the headings in the different sources, even in a same source, different news usually have a different format from the heading.

The most important conclusion of these results is that using training data from other sources different from the test file one is quite useful, we will study this more deeply in section 6.4.

6.4 Extraction across news domains

Until now, all the predictions have been done using training data from the same source, totally or partially. Now, we want to see what happens with the predictions of pages from an unseen source. From a practical point of view, this is the most useful type of experiment. If you want to build a news search engine or a multi-document summarization system which covers most/all of the news sources on the web. It is virtually impossible to annotate training data from all of them. Not only would it involve a lot of manual work and money, new ones are also appearing all the time, and formats are changing slightly all the time. It is therefore highly desirable to have an extraction system that can be trained on data from a limited number of news sources, but still performs reasonably well on unseen data.

First because it is very interesting to know how the system works in this mode, how TiMBL can predicts the tags of news using training data rather different from the news. And second because if in the future we want to tag a page out of the 16 original sources, it is very interesting to know if it is possible without adding any training data from that source.

In the tables 6.7, 6.8 and 6.9 you can see the results for this experiment, since the training data used are from different sources, we didn't expected a good performance.

The first impression is that the results are not good, 39,0 % in the F-score of the title and 44,4 % in the heading. The only quite good results are the body ones, 86,3 % is a promising start, specially if we think that in some problems the title and heading are not important, for instance, in searching problems, where you are only interested in the content of the news.

It is important to notice that in this experiment, the k-parameter is very relevant, we used the default settings, i.e. k-parameter equal 1, so just the most similar files are being used to make the predictions. However, since there are not files from the same source of the test file, TiMBL bases its predictions on the direct neighbours only, which may be quite different. This would explain why the title and heading predictions are so bad. On the other hand, the results in the body predictions are explained because as we said, the most important features to predict the body are the length one and the dot one, and they don't depend on the source, these are common features.

Table 6.7: Results on title extraction across news domains

SOURCE	# FILES	prec(Δ)	recall(Δ)	fscore(Δ)
aljazeera	18	0.207(-0.793)	1.000(+0.000)	**0.343(-0.657)**
guardian	17	0.667(+0.183)	0.588(-0.294)	**0.625(+0.000)**
BBC	23	0.840(-0.117)	0.955(-0.045)	**0.894(-0.084)**
CNN	10	(nan)	0.000(-0.778)	**(nan)**
xinhuanet	20	0.409(-0.591)	0.900(-0.050)	**0.563(-0.411)**
latimes	10	1.000(+0.167)	0.100(-0.900)	**0.182(-0.727)**
nytimes	15	(nan)	0.000(-1.000)	**(nan)**
reuters	17	0.000(-1.000)	0.000(-1.000)	**(nan)**
theautralian	26	0.000(-1.000)	0.000(-1.000)	**(nan)**
timesonline	16	0.269(-0.731)	0.875(-0.125)	**0.412(-0.588)**
dw-world	10	(nan)	0.000(-0.900)	**(nan)**
fox	4	0.231(-0.769)	0.750(-0.250)	**0.353(-0.647)**
hindu	5	(nan)	0.000(-1.000)	**(nan)**
upi	11	0.500(-0.500)	1.000(+0.000)	**0.667(-0.333)**
voamerica	12	0.000(-1.000)	0.000(-1.000)	**(nan)**
washintonpost	8	1.000(+0.000)	0.875(-0.125)	**0.933(-0.067)**
macro-mean		0.320(-0.598)	0.440(-0.529)	**0.311(-0.627)**
micro-mean		0.334(-0.558)	0.468(-0.505)	**0.390(-0.540)**

Table 6.8: Results on heading extraction across news domains

SOURCE	# FILES	prec(Δ)	recall(Δ)	fscore(Δ)
aljazeera	18	0.958(-0.042)	0.852(-0.148)	**0.902(-0.098)**
guardian	17	0.000(-0.333)	0.000(-0.400)	**(nan)**
BBC	23	0.903(+0.079)	0.651(-0.326)	**0.757(-0.137)**
CNN	10	0.000	(nan)	**(nan)**
xinhuanet	20	0.000	(nan)	**(nan)**
latimes	10	0.000	(nan)	**(nan)**
nytimes	15	(nan)	(nan)	**(nan)**
reuters	17	(nan)	0.000(-0.714)	**(nan)**
theautralian	26	0.167(-0.728)	0.056(-0.888)	**0.083(-0.836)**
timesonline	16	0.062(-0.938)	0.429(-0.571)	**0.109(-0.891)**
dw-world	10	1.000	0.750	**0.857**
fox	4	0.000	(nan)	**(nan)**
hindu	5	(nan)	(nan)	**(nan)**
upi	11	0.000	(nan)	**(nan)**
voamerica	12	0.000	(nan)	**(nan)**
washintonpost	8	(nan)	0.000(-1.000)	**(nan)**
macro-mean		0.193(-0.672)	0.171(-0.691)	**0.169(-0.690)**
micro-mean		0.400(-0.475)	0.500(-0.438)	**0.444(-0.461)**

Table 6.9: Results on body extraction across news domains

SOURCE	# FILES	prec(Δ)	recall(Δ)	fscore(Δ)
aljazeera	18	0.964(-0.030)	0.970(-0.018)	**0.967(-0.024)**
guardian	17	0.447(-0.515)	0.969(-0.008)	**0.612(-0.357)**
BBC	23	0.983(-0.010)	0.981(+0.026)	**0.982(+0.008)**
CNN	10	0.989(+0.034)	0.915(-0.050)	**0.950(-0.010)**
xinhuanet	20	0.896(-0.071)	0.987(-0.006)	**0.939(-0.041)**
latimes	10	0.975(+0.032)	0.752(-0.215)	**0.849(-0.106)**
nytimes	15	0.952(-0.030)	0.896(-0.097)	**0.923(-0.064)**
reuters	17	0.682(-0.300)	0.524(-0.476)	**0.593(-0.398)**
theautralian	26	0.859(-0.130)	0.927(-0.049)	**0.892(-0.090)**
timesonline	16	0.761(-0.222)	0.880(-0.072)	**0.816(-0.151)**
dw-world	10	0.452(-0.548)	0.875(+0.375)	**0.596(-0.071)**
fox	4	0.816(-0.153)	1.000(+0.000)	**0.899(-0.085)**
hindu	5	1.000(+0.021)	0.980(+0.078)	**0.990(+0.051)**
upi	11	0.797(-0.185)	0.965(-0.009)	**0.873(-0.105)**
voamerica	12	1.000(+0.016)	0.786(-0.182)	**0.880(-0.096)**
washintonpost	8	0.897(-0.085)	0.955(-0.027)	**0.925(-0.057)**
macro-mean		0.842(-0.136)	0.898(-0.045)	**0.855(-0.100)**
micro-mean		0.823(-0.157)	0.908(-0.061)	**0.863(-0.112)**

Another reason that explains the good results in the body prediction is that there are many more examples of body than of title or heading to learn from in every article used in the training data. So the amount of training data for this particular text element is bigger.

In the next section we will study the influence of the k-parameter and we will see if our suppositions are in the right track.

6.5 Feature weights using Information Gain

As we explained in chapter 3, Information Gain (IG) measures how much a feature contributes to our knowledge. In table 6.10 you can see the feature weights calculated by TiMBL using IG in the three kind of experiments we are running. To calculate these values, TiMBL runs a pre-process analyzing the training data, since the training data for the 3 kind of experiment come from the same pool (in each experiment we use different parts of the pool), we expected that average weights were almost the same in the 3 experiments.

This is exactly what we can see in table 6.10.

As we expected the length features are the most important, the current ones (number 15 and 22) have the higher weight among all of them. The current feature is the one with high weight in each group (Html structure, length ...),the values decrease when it moves away from the current. Is is interesting to see the weights for the tail length features. Features 21 and 23 have less weight than 20 and 24 even if they are closer to the current. Studying the html structure of the news articles we realized that the sources that use the tail of the html elements as main part to introduce the body never use the previous element and the next one, this could explain this effect.

6.6 Optimization of classifier settings

The performance of TiMBL depends on 4 aspects:

i. Which features are you adding.

ii. How much training data are you using.

iii. Which parameters and configuration is TiMBL running with.

iv. The nature of the problem.

To improve the results of the predictions we had to look into the three first aspects. The feature system is explained in chapter 5, while the amount of training data will be investigated in section 6.7. In this section we will study the configuration of TiMBL, until now, all the experiments were run with the default settings of TiMBL, i.e. IB1 algorithm, feature weighting with Gain Ratio, weighted overlap and number of nearest neighbours (k-parameter) equal 1.

It is important to notice that these default settings were chosen by TiMBL designers because they are good in most of the problems, so we are not too much optimist about improving the results a lot.

To find the best configuration of k-parameter and feature weighting (w-parameter), and to know which was the best algorithm to this problem, we did an exhaustive search of the parameter space (within certain limits).

Table 6.10: feature's weight average from extraction in the three kind of experiments

	Html structure features	single	merging	new
1	element -5:	0.012	0.012	0.012
2	element -4:	0.014	0.014	0.014
3	element -3:	0.016	0.017	0.016
4	element -2:	0.019	0.019	0.019
5	element -1:	0.022	0.021	0.022
6	**current element:**	0.030	0.030	0.030
7	element +1:	0.023	0.023	0.023
8	element +2:	0.021	0.021	0.021
9	element +3:	0.017	0.017	0.017
10	element +4:	0.016	0.016	0.016
11	element +5:	0.013	0.013	0.013
	Text length features			
12	element -3:	0.083	0.082	0.082
13	element -2:	0.105	0.105	0.105
14	element -1:	0.119	0.118	0.118
15	**current element:**	0.206	0.205	0.205
16	element +1:	0.119	0.118	0.118
17	element +2:	0.098	0.098	0.098
18	element +3:	0.078	0.077	0.077
	Tail length features			
19	element -3:	0.013	0.014	0.014
20	element -2:	0.122	0.121	0.121
21	element -1:	0.010	0.011	0.011
22	**current element:**	0.223	0.221	0.222
23	element +1:	0.010	0.011	0.010
24	element +2:	0.137	0.136	0.136
25	element +3:	0.011	0.011	0.011
	Letter features	ratios		
26	text letter:	0.051	0.052	0.052
27	tail letter:	0.101	0.101	0.101
	Text dots features			
28	element -3:	0.035	0.035	0.035
29	element -2:	0.045	0.045	0.046
30	element -1:	0.056	0.056	0.056
31	**current element:**	0.100	0.100	0.100
32	element +1:	0.056	0.056	0.056
33	element +2:	0.048	0.048	0.048
34	element +3:	0.036	0.036	0.036
	Tail dots features			
35	element -3:	0.008	0.012	0.008
36	element -2:	0.065	0.005	0.065
37	element -1:	0.008	0.014	0.008
38	**current element:**	0.135	0.040	0.135
39	element +1:	0.008	0.008	0.008
40	element +2:	0.078	0.077	0.077
41	element +3:	0.008	0.008	0.008

We run three experiments as before (extracting within single news domains, merging domains and across news domains) for each combinations of algorithm, k-parameter and w-parameter.

In our search we test the following values for the parameters:

- We tested 2 algorithm from very different families, IB1 and IGTREE.
- the k-parameter from 1 to 10.
- w-parameter with all its values, i.e. No Weighting, Weight using GainRatio (default), Weight using InfoGain, Weight using Chi-square and Weight using Shared Variance.

We decided to stop in 10 for the k-parameter because we observed that the performance was decreasing after k-parameter equal 7 in all the cases as you can see in the section 6.6.1, so it was not worthwile trying with higher values.

6.6.1 Number of nearest neighbours (k-parameter)

The relevance of k-parameter depends on which kind of training data are being used, if all the data comes from the same domain, such as the experiments of section 5.2, the similarity of the data makes bigger values of k-parameter useless, at least, this was our prediction. In section 6.4 we talked about the effect of the k-parameter could have on the performance of the predictions when there is no training data from the same source as the test file. We think that it is more interesting to see what happened with k-parameter in this situation than in the previous one.

It is difficult to study the effect of a single parameter when the results depend on others. To understand how the performance is modified by k-parameter we calculated the average of the F-score of all those experiments grouped by k-parameter. For instance, if we name the results files as *res-A-K-W* where A is the algorithm used, K the k-parameter and W the weighting features (from 0 to 4), to know the performance of the predictions with k-parameter equal 3, we made the average of the F-score from all the files with K equal 3. We were also interested on know if the behaviour of the F-score while k-parameter changes is the same for the three parts of the news (title, heading and body) or not, so in the next graphics we will show the evolution of F-score independently of the title, heading and body.

The graphic 6.1 shows how the F-score descends with higher values of k-parameter in the title, heading and body. This is exactly what we expected. This time the default value was the best one.

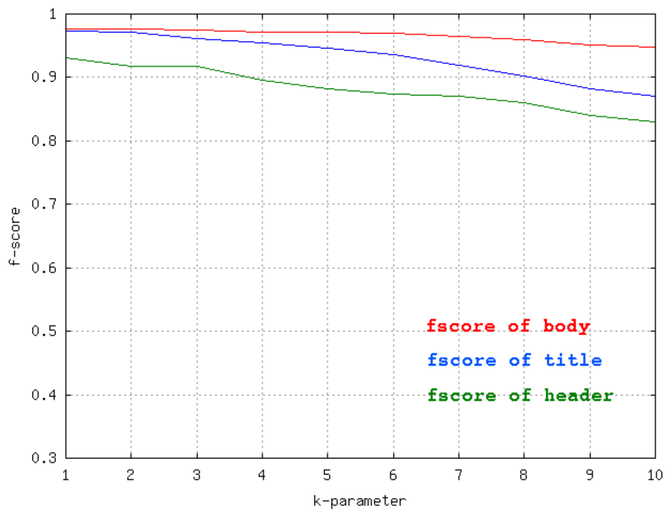


Figure 6.1: F-score in title, heading and body as a function of parameter k, with merged domains

In graphic 6.2 you can see that the evolution of the F-score when we make the extraction across new domains is totally different from the previous one, k equal 1 is not the best one anymore. Although there are some variations, we can see that around k equal 7 the F-score get a global optimal value.

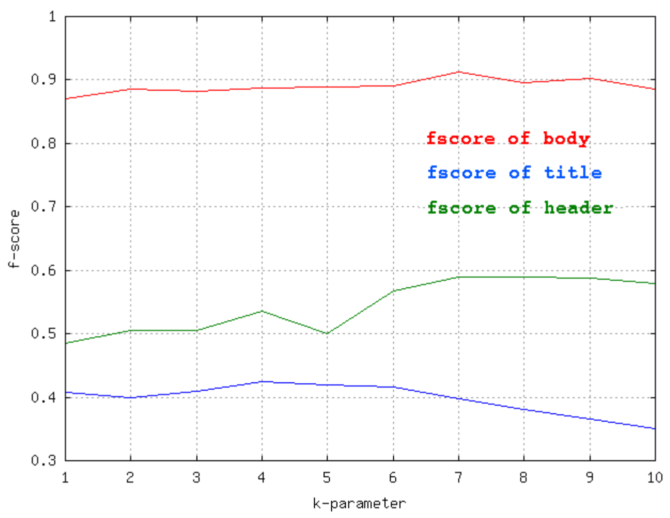


Figure 6.2: Evolution of F-score in title, heading and body as function of parameter k, across new domains

6.6.2 Feature weighting (w-parameter)

To understand the relevance of feature weighting we did the same as with k-parameter, we calculated the average of the F-scores grouped, this time, by w-parameter. This time, we did not have any prediction of which weight would be better or how the kind of experiment (merging domains or across new domains) could be affected by w-parameter. In table 6.11 you can see the global F-score in the title, heading and body by w-parameter in the extraction merging domains.

Table 6.11: global F-score in the title, heading and body with all values of feature weighting, using merging domains

Feature weighting	F-score			
	title	heading	body	average
No-Weighting	0.958	0.917	0.972	0.949
GainRatio	0.961	0.902	0.976	0.946
InfoGain	0.968	**0.923**	0.972	0.954
Chi-square	0.964	0.917	0.975	0.952
Shared Variance	**0.973**	0.917	**0.979**	**0.956**

Even if the InfoGain is better to predict the heading, the best feature weighting in overall is the Shared Variance.

Table 6.12: global F-score in the title, heading and body with all values of feature weighting using across new domains

Feature weighting	F-score			
	title	heading	body	avegare
No-Weighting	0.308	0.543	0.898	0.583
GainRatio	0.316	0.539	0.874	0.576
InfoGain	0.306	**0.594**	**0.901**	0.600
Chi-square	**0.577**	0.523	0.893	**0.664**
Shared Variance	0.443	0.414	0.744	0.533

Using all the training data, i.e. with the experiment across new domains, there is a lot of variability as you can see in the table 6.12 but we can say that Chi-square is on average the best one.

6.6.3 Optimal settings

In this section, we apply the optimal combination of k-parameter and w-parameter with the algorithm that better results has obtained, IB1 (to see the results with the IGTREE see section 6.6.4). So these are the best results we could have obtain running all the experiments. It is important to notice, that these results are the best ones predicting the 3 parts of the news. It is possible to find a better setting for a specific part as we showed in the previous sections.

We made the optimization for two kind of experiments, using all the training data that we have (extraction merging domains) and using as training data just the articles from different sources than the test file (extraction across new domains). With the first kind of experiments, the optimal results were obtained using as algorithm IB1, k-parameter 1 and feature weighting Shared Variance, while with the second kind the optimal parameters were algorithm IB1, k-parameter 7 and feature weighting Chi-Square.

In table 6.13 you can see these optimal results compared with the results obtained with the default settings of TiMBL.

In the case of merging domains, since the k-parameter and the algorithm are the same in the default settings than in the optimal one, there is not a big difference between them. We just can noticed an interesting improvement predicting the heading. If we calculated the error reduction, going from 0.085

Table 6.13: Results with optimal and default parameters

SOURCE	TP	FP	TN	FN	prec	recall	fscore
				TITLE			
optimal (merging domains)	214	4	138271	7	0.982	0.968	**0.975**
default (merging domains)	212	2	138273	9	0.991	0.959	**0.975**
optimal (across new domains)	153	128	138147	68	0.544	0.692	**0.610**
default (across new domains)	103	206	138070	118	0.334	0.466	**0.369**
				heading			
optimal (merging domains)	110	7	276869	6	0.940	0.948	**0.944**
default (merging domains)	107	11	276865	9	0.907	0.922	**0.915**
optimal (across new domains)	64	69	276807	52	0.481	0.552	**0.514**
default (across new domains)	58	87	276789	58	0.400	0.500	**0.444**
				BODY			
optimal (merging domains)	3312	62	412049	65	0.982	0.981	**0.981**
default (merging domains)	3303	52	412059	74	0-985	0.978	**0.981**
optimal (across new domains)	3049	408	411703	328	0.882	0.903	**0.892**
default (across new domains)	3066	656	411455	311	0.824	0.908	**0.864**

(1-0.915) to 0.056 (1-0.944) represents a 51.78 % of improvement. However, the optimal settings for the experiment across new domains make the results much better, the main reason is the use of k-parameter 7, that as we saw in graphic 6.2 has a big relevance. If we calculate again the error reduction, we can see a relative improvement of 61.8% in the title, 14.4% in the heading and 25,9% in the body.

6.6.4 Algorithm

As we explained in chapter 3, the IGTREE algorithm is usually faster than IB1 but the performance is usually worse. In this specific problem, where the amount of data is not too big and the processing time is not the problem, using IGTREE is not worth because the results are quite worse.

We did the same parameter optimization that we explained for the IB1 algorithm in section 6.6 but now, using IGTREE (without k-parameter since IGTREE is not affected by the number of nearest neighbours) and we found that the optimal w-parameter was 0 (without feature weighting) to both, merging domains and across new domains. In table 6.14 you can see the best results (optimal) obtained with both algorithms in the two kinds of experiments. As you can see the best results using IGTREE are far from the best results using IB1.

Table 6.14: Results with optimal and default parameters

SOURCE	TP	FP	TN	FN	prec	recall	fscore
				TITLE			
IB1 (merging domains)	214	4	138271	7	0.982	0.968	**0.975**
IGTREE (merging domains)	209	3	138272	12	0.986	0.946	**0.965**
IB1 (across new domains)	153	128	138147	68	0.544	0.692	**0.610**
IGTREE (across new domains)	124	135	138156	81	0.479	0.605	**0.534**
				heading			
IB1 (merging domains)	110	7	276869	6	0.940	0.948	**0.944**
IGTREE (merging domains)	80	6	276870	36	0.930	0.690	**0.792**
IB1 (across new domains)	64	69	276807	52	0.481	0.552	**0.514**
IGTREE (across new domains)	51	84	276804	53	0.378	0.490	**0.427**
				BODY			
IB1 (merging domains)	3300	87	412024	77	0.974	0.977	**0.976**
IGTREE (merging domains)	2902	500	411611	475	0.853	0.859	**0.856**
IB1 (across new domains)	3049	408	411703	328	0.882	0.903	**0.892**
IGTREE (across new domains)	2798	687	411426	577	0.803	0.829	**0.816**

6.7 Effect of data size

We have studied the effect of different parameters and different kind of training data on the accuracy of the predictions done by TiMBL, but in all these experiments we were using the same pool of training data, in this section however, we are going to study the effect of data size on the TiMBL performance.

We wanted to measure how much sensitive is TiMBL to the data size. So we designed the following experiment: We started with a pool of training data of 2 articles, we ran a cross-validation and we calculated the global F-score as an average of the F-score obtained in the title, heading and body. Then, we added a third article and we did the same, calculating the new F-score. We repeated this until we finished all the training data we had. The graphic that shows the relation between the number of files used to run the cross-validation and the global F-score obtained is called learning curve.

Since we have 3 kind of extractions (from a single domain, merging domains and across new domains), we designed 3 sets of experiments. All these experiments were done using the optimal settings (calculated in the previous section). In the next sections, 6.7.1, 6.7.2 and 6.7.3, the learning curves are shown and analyzed.

6.7.1 Learning curves from single news domain

This section is less interesting than the next ones since we are running a set of cross-validations for each source, the amount of training data used in one is not very big (around 20 files), so you can not see the learning curve very well. Anyway, we can take some interesting conclusions.

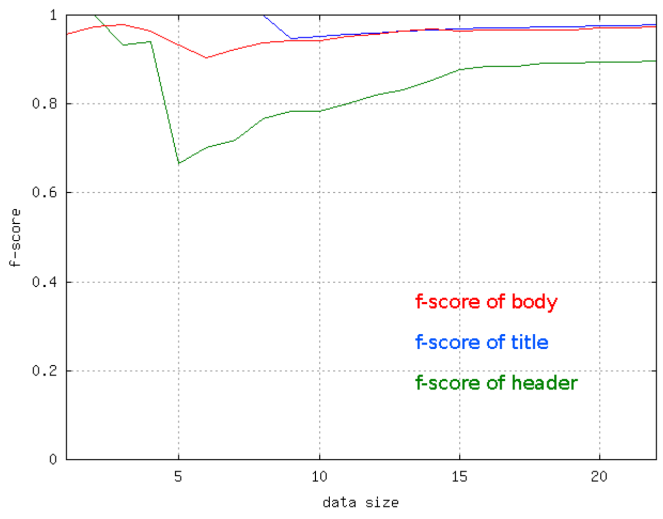


Figure 6.3: BBC learning curve

In the first learning curves of the graphics 6.3, 6.4 and 6.5 we can see an overall improvement in the F-score when we increase the number of training files. However, at the beginning in some of them , there is a worsening before the improvement. When there is a small training data, the performance depends highly on the similarity of the files we are using, so the results, when small sets of training data are used, change a lot from one source to other, all deppending on the similarity.

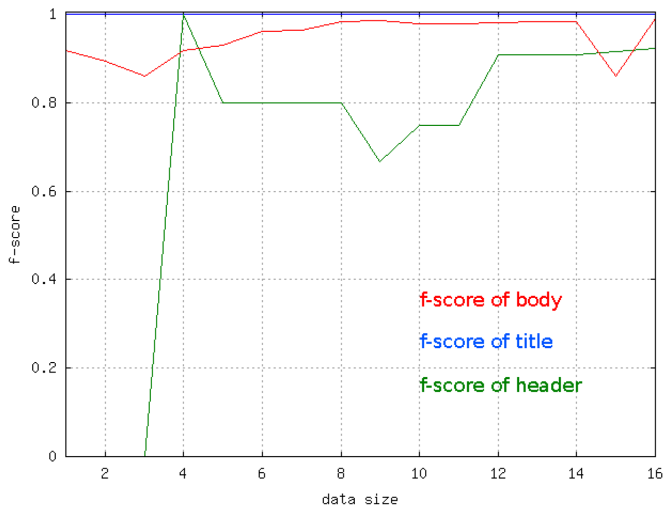


Figure 6.4: Reuters learning curve

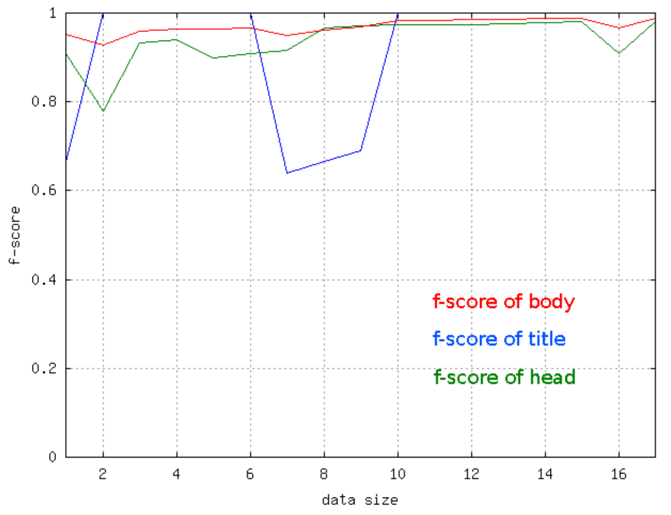


Figure 6.5: Aljazeera learning curve

Another interesting example is graphic 6.6. Here we can see a "black sheep", the last files of The Australian were in a different format than the others, so, when we add the first these "weirds", there is not any similar file that TiMBL can use to predict it, so the F-score descends. When we add more training data in a similar format than the "black sheep" the F-score improves again.

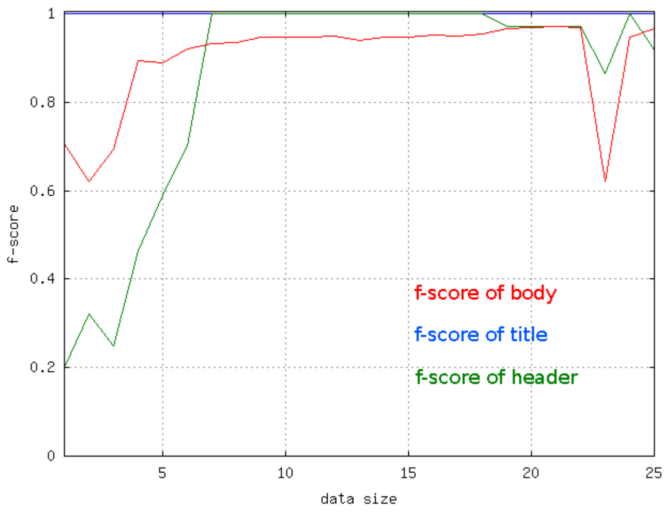


Figure 6.6: The Australian learning curve

6.7.2 Learning curves merging domain

For this experiment all the training data we had were used, from 2 files until 222 files, we run a set of cross-validations adding randomly the new files.

This is the most meaningful learning curve, in graphic 6.7 you can notice perfectly the improvement in the F-score while increasing the number of files.

Since we are adding one by one files from different sources, at the beginning TiMBL is trying to predict each files using as training data a small pool of articles from different sources. This makes the graphic oscillate quite a lot when there are not many files

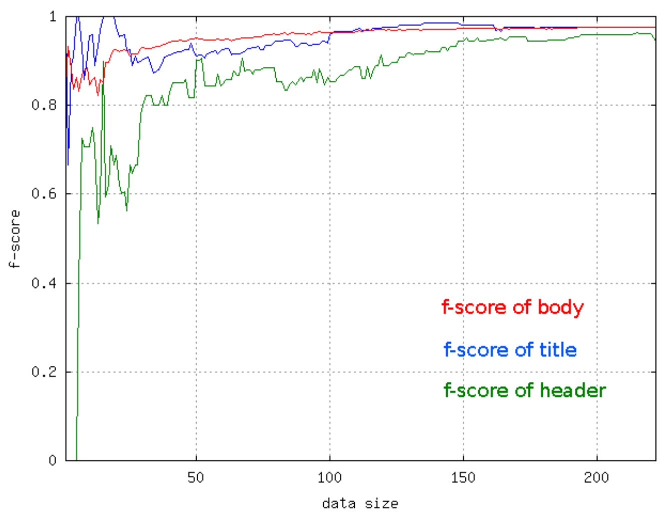


Figure 6.7: Learning curve from extraction merging domains

From 25 to 150 files, the graphic rises almost constantly, this is the most important conclusion and it proves the sensitivity of TiMBL to the data size. However, there is one point (around 150 files) where the F-score stops to increase or increase very slowly and the curve becomes asymptotic. The meaning of this is that all the files we are adding since this point have an already known structure/format, so no new knowledge is being adding and the F-score doesn't change.

The point where the F-score stops to increase is very important from a practical point of view. It means that from this point we are not adding new information to the learning system, so all the training data in the same format as before is useless. We will talk more about this and how to avoid this effect in section 6.8

6.7.3 Learning curves across news domain

The learning curve this time, graphic 6.8 is quite similar to the previous ones, you can see an improvement on the performance with the increase of data size.It is important to notice that due to the kind of experiment (without using training data from the same source) we are representing the prediction by source, not by article, i.e., each new training file added is all the articles from a news source, so we can not see what happened one by one.

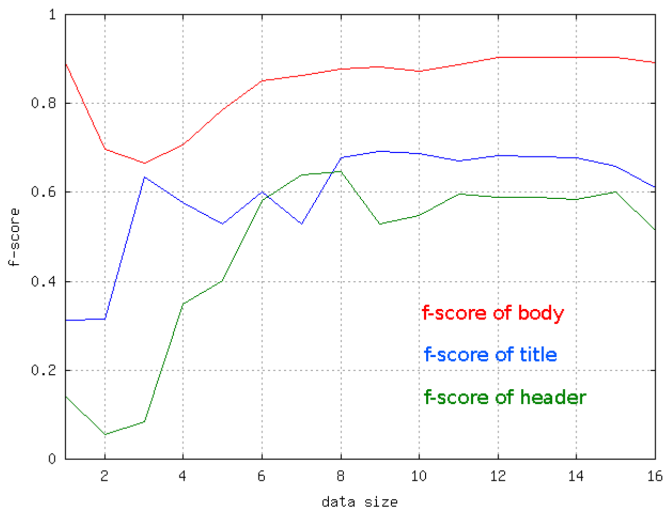


Figure 6.8: Learning curve from extraction across news domain

6.8 Post-processing

In this section we will talk about the post processing systems that we made, ie, treatment of the data after the TiMBL execution.

The advantage of this kind of treatment is that we can use a global knowledge, all the class predictions done by TiMBL are independent of each other. For example, when predicting the class of a certain instance, TiMBL does not know the prediction for the preceding or following instances. But now, since these processes are run after TiMBL execution, we know all the class predictions of the news article, so we can use new global features to write new scripts that let us evaluate the quality of these predictions.

The global feature we are going to use is the global structure of the article, i.e. the sequence of title, heading and body. It is important to notice that we could not use this before, since we could not know the global structure until TiMBL has predicted it. The idea is to identify those articles tagged by TiMBL that have ill-formed structures, specifically we want to identify these three errors:

i. News article with no title (E1).

ii. News article with more than 1 title (E2).

iii. News article with 2 or more headings together (E3).

We made a state machine, as you can see in picture 6.9. This machine has the original state, 3 states for title, heading and body and 3 final states, one for each kind of error.

We wrote this state machine into a script and we found the errors shown in table 6.15, over 222 news articles we found 10 global structure errors, as you can see 9 of the 10 errors are due to the absence of title and the other one because TiMBL predicted 2 headings together.

Table 6.15: Errors found using post-processing

NEWS ARTICLE	**ERROR**
www.dw-world.de-1-1303703710	**no title**
www.guardian.co.uk-6-1304190973	**no title**
edition.cnn.com-1-1304190973	**no title**
www.abs-cbnnews.com-1-1303999305	**no title**
news.bbc.co.uk-3-1303503657	**2 headings together**
edition.cnn.com-2-1304190973	**no title**
www.latimes.com-4-1304190973	**no title**
www.voanews.com-1-1303703710	**no title**
news.bbc.co.uk-1-1303703710	**no title**
www.guardian.co.uk-3-1303843291	**no title**

Thanks to this post-processing treatments, like the previous one, we can have a kind of metric of the predictions, at least we can know which predictions are wrong in determined sense, for instance, we can know which

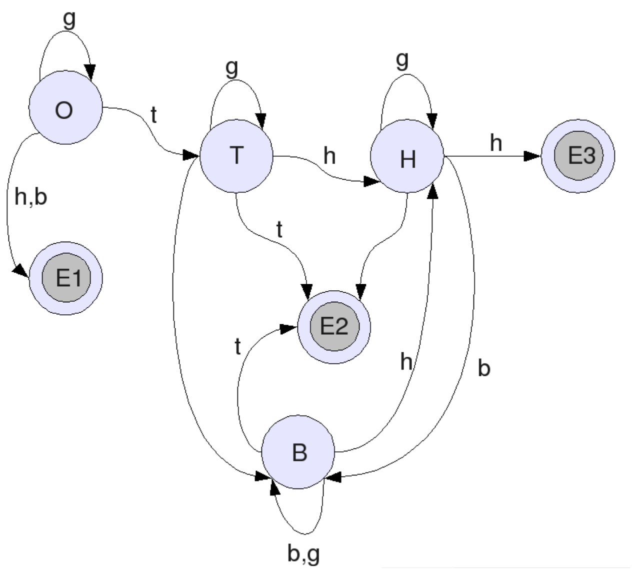


Figure 6.9: State machine that detect ill-formed structures

articles are badly predicted from the structure point of view. In section 6.7.2 we analyzed the effect of data size and we talked about the uselessness of adding, as new training data, news articles that TiMBL predicts quite well. The idea is that if TiMBL can predict these articles with high accuracy is because there are already another ones in the training data with a similar structure and format, so, the knowledge of the training system is not going to improve. If we want to extend the knowledge we need to add articles in a different format or structure, and it means, articles that TiMBL can not predict correctly. But to know when the predictions of TiMBL are good or not, without asking humans to check it, we need some metrics of the predictions, and this is exactly what we have with this post processing treatment.

Chapter 7

Discussion and conclusion

As presented in chapter 3 there are several approaches for automatic wrapper induction. In this thesis we have studied the performance of a specific supervised classifier, namely a memory-based learner.

We have used the same pool of training data in three different ways: within single news domain, merging domains and across new domains. The differences in the global performance obtained with these three kind of experiments reveal that the similarity between the training data and the test data is very important, as is to be expected. If we could have a system up to date with training data from all the sources we are interested in extracting information from, the predictions could be almost perfect. However, we know that this situation is unrealistic because it not only would involve a lot of manual work and money, new sources are also appearing all the time, and formats are changing slightly all the time.

We also have shown that the data size is an important factor in the performance. Even adding data from different sources, helps in the prediction as we saw in section 6.3. It is important to notice that as we saw in figure 6.7, increasing the training data is useful as long as we are adding examples which have not been seen so far. The MBL approach implies that it is useless to add training data with similar structure/format once we have reached the asymptotic point shown in graphic 6.7. If the majority of the examples in our training set match with the test file, adding a new example that match also, does not produce a significant improvement on the performance.

We took all the training data from English sources, but after the features construction we realized that we did not use any feature related specifically to the English language, so this system can be extended to any other language.

Another point of discussion is the consistency of the annotation, as we said in section 4.1.1. There could be disagreements labeling the news articles, for instance, what is a heading and what is not. This could change the results shown in the same way. So, the way to make the annotation could be a discussion point if, some day, we wanted to extend this thesis and use more than one person to annotate the data.

We think that the same approach can work for similar extraction from web pages problems. The features we used to classify the elements of the news article can be divided into html structure features and format features. This kind of features could also identify the significant data for many mining-web problems like prices-mining, mining emailaddresses or telephone numbers etc. This kind of problems could be studied following the same procedure used in this thesis.

7.1 Future work

Although the experimental results are very encouraging, there are still some issues that deserve further research.

One behaviour we have noticed during the optimization of the settings (section 6.6) is that different text elements have different optimal feature weighting schemes. This suggests that we can obtain further improvement by building separate classifiers for each prediction of the three text elements, and combining them using a meta-classifier.

The performance results and the learning curves obtained in the experiment across new domains (sections 6.4 and 6.7.3) suggest that the performance on the heading and title could be improved by adding more training data. As we said, one of the reasons that makes the body performance higher is that in each news article there are more examples of body than title or heading, so we could say that there is more body training data than heading or body. Adding more examples is a very interesting future work.

During the construction of the learning curves shown in section 6.7 we detected an inherent problem: the order we use to add the training files. Imagine that you have 3 files, A, B and C. Using as training data the file C you can predict with 100 % of accuracy files A and B, but A and B are quite bad files used as training data. If we run our experiment adding them in the order A -> B -> C we will have worse results than if we add them in the order C -> A -> B. With this little example we want to show that the learning curves are conditioned by this effect. Anyway, this effect is quite important with a small set of files, the most files we use, the less this effect is relevant. One way to fix it could be to run several times the same experiment we did to make the learning curves, but adding extra annotated files in a random order (rather than in a fixed order, as was done here). The average of this executions would show us a more realistic learning curve.

We are also interested in continue work on post processes, as we said in section 6.8. With this kind of treatment we can obtain a measure of the quality of the predictions. In our problem, the unlabeled data are abundant, but labeling them is time consuming. So it could be useful to label the raw data with our learning system and use this quality metric to detect the examples badly labeled by the system. We can focus our human resources on labeling these examples that really add knowledge to the system. This type of iterative supervised learning is called active learning.

Even though there are many issues that require further research, we consider our attempt to rephrase automatic wrapper induction as a general memory-based classification problem to be succesfull.

Chapter 8

Refences

- Walter Daelemans and Antal van den Bosch [2005]. Memory-based language processing.
- Bing Liu [2007]. Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.
- Davi Castro, Reis Paulo, B. Golgher and Altigran S. Silva. Automatic Web news extraction using tree edit distance. Proceedings of World Wide Web Conference (WWW04).
- Pierre Senellart, Avin Mittal, Daniel Muschick, Remi Gilleron and Marc Tommasi, Automatic Wrapper Induction from Hidden-Web Sources with Domain Knowledge.
- Ian H. Witten and Eibe Frank [2000]. Data Mining: Practical machine learning tools and techniques with java implementations.
- Hahn, U. and Mani, I. [2000]. The challenges of automatic summarization