MASTER IN COMPUTER SCIENCE

Use of Web mining for an actualized and coherent chatterbot dialogue

Dosquet, Benjamin; Magnant, Xavier

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année académique 2003-2004

Use of Web mining for an actualized and coherent chatterbot dialogue

Benjamin Dosquet & Xavier Magnant

Mémoire présenté en vue de l'obtention du grade de Maître en informatique

Résumé

Dans une époque ou de plus en plus d'interactions s'opèrent dans le monde virtuel au lieu du monde réel, les dialogues gagnent en anonymat, et de nouvelles opportunités d'interactions apparaissent. Le chatterbot est une de ces opportunités. Un chatterbot est un programme capable de dialoguer avec un humain en simulant une conversation. Tout bon chatterbot se doit d'être crédible. Pour être crédible, une série de conditions sont nécessaires. Il doit parler d'une manière relativement cohérente, tout en étant capable de donner son avis sur des sujets divers et actuels.

Dans ce mémoire, nous présenterons les outils qui permettent d'atteindre ces objectifs, en particulier l'intérêt d'Internet et des théories émanant du web mining et du text mining. Nous présenterons également le cheminement de l'implémentation de certaines techniques dans un chatterbot concret, qui ajoute la particularité de détecter des personnes donnant des conseils boursiers dans des salons de discussion publics, avant d'entamer un dialogue avec eux.

Mots-clés: chatterbot, data mining, web mining, text mining.

Abstract

In an era where more and more interactions between people or companies are taking place in the virtual world instead of the real world, there is an opening for anonymity and at the same time an opportunity for new types of interactions. The chatterbot is one of those opportunities. A chatterbot is a computer program that is able to hold a dialogue with humans by simulating a conversation. A good chatterbot has to be believable. To be believable, a set of conditions are necessary. It has to speak in a coherent way, has to be able to give its opinion on some subjects, and has to be aware of current events.

In this thesis, we will present some tools which allow to achieve these aims. We will especially see the interest of the internet and some text and web mining theories. We will also present an ordered framework of the techniques to implement into a concrete chatterbot, which adds the particularity to detect people giving advice about stock exchanges in a public chatroom, before starting a dialogue with them.

Keywords: chatterbot, data mining, web mining, text mining.

Foreword

This thesis is based on our resident internship in the University of Technology, in Sydney, Australia and was partially presented at the IASTED international conference on Artificial Intelligence and Applications (AIA 2004) in Innsbrück, Austria.

We especially wish to thank our promoter, Mrs Noirhomme-Fraiture, and our internship master, Mr Simeon Simoff for their help. We also have to thank Debbie Zang and Paul Bogg, two Australian students who worked in the same research group, as well as François Barthélemy and Samuel Gries, two Belgian students who worked on the same project.

Table of contents

Introduction

In this thesis we'll try to explain how web information, text mining techniques, and web mining techniques can be useful to a chatterbot. We'll try to show that they allow a chatterbot to speak in an actualized and coherent way.

Actually, a chatterbot can be defined as a computer program able to converse with humans by simulating a dialogue. Thus, one of the main aims of a chatterbot is to speak about current events, and general truths.

If a chatterbot only knows what the speaker tells him, or if it would talk only about things which happened in the past, it wouldn't be believable. We'll see how web mining techniques allow to achieve this aim.

Of course, a chatterbot also has to speak in a coherent way. If it spoke nonsense all the time, it wouldn't be believable either. We'll see how this aim can be achieved with text mining techniques.

The World Wide Web has turned into one of the most important sources of information in the world. A chatterbot can thus fetch eclectic information, and then use it as a source of knowledge to hold a dialogue.

Then we'll see a concrete application of the use of those web and text mining techniques in a chatterbot called the "Bogus Adviser Detection Chatterbot". This bot was developed within the framework of the e-market research group of the University of Technology Sydney, Australia. Basically, it simply holds a dialogue with a human chatter and adds bogus advice detection in its specialised form.

This thesis is organized in three chapters. In the first chapter, we will give a global overview of what text mining is and what it can do. We will describe the main text mining techniques, and point out which are really useful to the chatterbot and which are not.

The second chapter is focused on the concrete use of web mining techniques in the case of a chatterbot. We'll see how existing web mining techniques are useful to make a chatterbot aware of current events.

The third and last chapter will describe our application of a chatterbot, i.e. how it works, and which of the previously described techniques are actually implemented in our program.

Chapter 1: Text Mining

Too much knowledge doesn't facilitate simple decisions
Frank Herbert, Dune

1. Introduction

This chapter's purpose is to show what the most used text mining tasks are, what they involve, and which theories they are based on.
The hypothesis of this chapter is that the texts to be processed have already been selected.
The chapter is divided in four main sections:

- *"Textmining defined"* will focus on the definition of text mining as well as the main reasons why text mining becomes more and more important.

- *"Preprocessing phase"* will focus on the methods that are used to transform pure text into something easier to handle by a computer. You may already note that most of the techniques described in this section are useful to a chatterbot.

- *"Processing phase"* will focus on the main techniques that text mining offers. This section ends with some thoughts about the usefulness of the techniques if they are applied to a chatterbot.

- *"WordNet"* will focus on a tool that adds a semantic dimension to text mining modules.

2. Text mining defined

We will see where text mining exactly lies within the KDD process and standard data mining. Let's define the KDD process at first.

KDD stands for Knowledge Discovery in Databases and the well-known definition from Fayyad states: "*KDD is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*" [Fayyad U., Shapiro G., and Smyth P. 1996]. The KDD process can be divided into different stages. We will define where general text mining lies in relation to the complete KDD process. Note that the sub-processes composing the whole process allow a refinement with iterations, which is expressed by the returning arrows on Figure 1.1 taken from [Hamilton H. Gurak E., Findlater L., and Olive W. 2004].

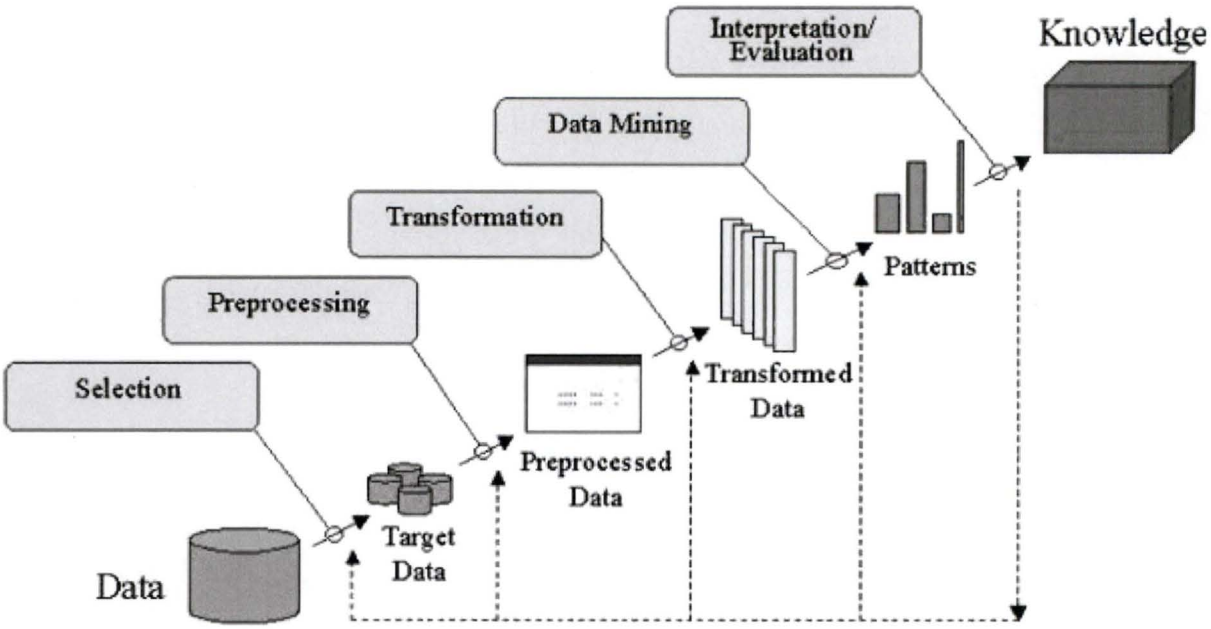


Figure 1.1 : KDD steps

More importantly, we notice that Data Mining is just a step in the KDD process.

The Data Mining step refers to the application of algorithms designed to extract patterns from data without the additional steps of the KDD process.

As a reminder, we sometimes say that data mining can be viewed as the detection of trends or patterns in huge amounts of data stored in databases.

Text mining is an extension of classical data mining. It fits in a KDT process (knowledge discovery in textual databases), right between the prior selection, pre-processing and transformation phases and the evaluation of the obtained results.

2.1 Explosion of Text Mining

Why is text mining becoming such a popular discipline? Let's go back to matter-of-fact facts about today's information society.

In the recent years, we have witnessed an impressive growth of the availability of information in electronic format. This is mostly due to the explosion of the Internet, but also to the presence of textual data warehouses in every field of our everyday life. Texts are more and more converted in electronic form and distributed in this practical format instead of a physical format.

When visiting Google's main page, we will notice the number of unique pages referenced by Google. As of may 2004, there are 4,285,199,774 pages of text of all nature. In 2000, a study reveals that there are "only" 2.1 billion unique publicly available pages on the Internet and that the Internet is growing at an explosive rate of more than 7 million pages each day, indicating that it will double in size by early 2001. [Murray B. H. and Moore A. 2000]

If we add the information that is contained in the form of emails and electronic texts in digital libraries or within businesses and governments, and we understand that the overwhelming amount of text brings an information overload problem.

Text Data Mining, also called Text Mining (TM for short) will deal around this information overload problem and will turn information that is buried in text into valuable knowledge.

Broadly speaking, text mining is similar to data mining, but instead of detecting patterns and trends on clean structured databases, it handles written texts and gives birth to previously hidden, implicit or unknown information, it classifies texts, summarizes them or clusters them.

A more formal definition extends the KDD definition: TM is the non trivial extraction of implicit, previously unknown, and potentially useful information from large amounts of textual data.

But what is something "previously unknown?"

There are two different views of the matter:

1. According to Hearst

Text mining is different from what we're familiar with in web search. In search, the user is typically looking for something that is already known and has been written by someone else. The problem is pushing aside all the material that currently isn't relevant to your needs in order to find the relevant information.
In text mining, the goal is to discover heretofore unknown information, something that no one yet knows and so could not have yet written down. [Hearst M.A. 2003]

Without mentioning the name, Hearst refers here to information retrieval and all kinds of search engines that we might encounter on the web. Information retrieval is not text mining, text mining is not information retrieval. One can borrow techniques from the other and the frontier sometimes disappears for that reason.

2. The second view adopted by many others

Text mining refers to the rediscovery of the information that the author buried in his texts. This may also be called previously unknown information, because a quick look at the text will not reveal the information explicitly. [Hidalgo J. 2002]

Those two views are different indeed. This thesis will not take the extreme view of what Hearst calls Real Text Mining, but it will discuss the most known text mining techniques from the second view. Some disciplines that are closely related to text mining will also be covered when they bring something that might be helpful for a chatterbot (WordNet for example).

2.2 What is not Text Mining

Everything that is related to text and computers is not text mining. If we take Fayyad's point of view regarding previously unknown information, we can identify several acts that are not text mining. We take "unknown" from the point of view of the reader, not the author. This means we limit the definition as less as possible. Finding keywords in a text can be defined as previously unknown information from the point of view of the reader for example, whereas the author will have written the text around a few predefined keywords.
Some techniques that are not considered as being real text mining include:

- **Traditional queries** like SQL queries on a textual database, because this only retrieves already known information.
- **Information retrieval**: Given a source of textual documents and a text based user query, information retrieval finds a ranked set of documents that are relevant to the query. Intelligent information retrieval adds a solution to the problem of meaning (synonyms and ambiguity) and to the order of the words in the query. It also takes the authority of the source into account (IBM is more likely to be an authorized source then someone with a homepage). Information retrieval is not considered as being real text mining, it is however strongly related. IR is mostly used by search engines on web pages [Hearst M.A. 1999]. Information retrieval will be discussed in section 3 of chapter 2.
- **Web mining**: There is a difference between information on a webpage and classical texts. On a webpage, text possesses a certain structure defined by tables like a traditional text possesses chapters or paragraphs. The main difference is to be found in the structure of the WWW, where pages and textual information are structured with the help of hyperlinks. The analysis of relations between textual information linked by the hypertext structure is web mining. A web mining module can retrieve pure texts that will be the input of a text mining module. Web mining will be detailed in the second chapter of this thesis.
- **Information extraction**: Given a source of textual documents and a well defined limited text based query, information extraction finds sentences with relevant information, extracts the relevant information and ignores non-relevant information. The output is in a predetermined format. IE aims in other words at filling domain

dependant templates with data buried in text items, and it is not considered as being real text mining.

2.3 Properties of text

Marti Hearst identified the main properties of text and particularly why text is tough and easy to handle. [Hearst M.A. 1997]

- Text is though
  - There are countless combinations of subtle, abstract relationships among concepts
  - There are many ways to represent similar concepts (Hearst refers to synonymy and subtleties of natural language to express a concept)
  - Concepts are difficult to visualize
- Text is easy
  - Text is easy because a very simple algorithm can already produce very good results for several tasks
  - Text expresses very rich and fertile ideas
  - Text is highly redundant in bulk

2.4 Limitations of text mining

Understanding the limitations of text mining is an important phase of any research or work that is done around the subject.

The fundamental limitations of text mining are first, that we will not be able to write programs that fully interpret text for a very long time, and second, that the information one needs is often not recorded in textual form. If I tried to write a program that detected when a where a new word came into existence and how it spread by analyzing web pages, I would miss important clues relating to usage in spoken conversations, email, on the radio and TV, and so on. Similarly, if I tried to write a program that processes published documents in order to guess what will happen to a bill in Washington DC, I would fail because most of the action still happens in negotiations behind closed doors. [Hearst M.A. 2003]

3. Preparation phase

3.1 Introduction

Gerard Salton explained the need to pre-process texts in a certain way before doing anything relevant on it. Every text mining or information retrieval system available today has features that emanated from Salton's work [Salton G. 1989].
There is no doubt that there are many relations and common problems to both disciplines. Stop-word removal and stemming are amongst those issues.

3.2 Stop-word removal

Stop-words are words that from a non-linguistic point of view do not carry information. In every language, some words have meanings that never play a central role in a text. We call those words, stop-words. We could say that the purpose of those words is to link meaningful concepts.
Stop-words are of course natural language dependant.
In French for instance, examples are: *le, un, je, vous, pour, de, ne, contre, avec, après, aussi, pourtant, etc.*
We don't want to keep those words that are very important for human beings but completely useless for any machine processing the texts to discover meaningful knowledge.

Perfectly removing stop-words automatically without human intervention or thesaurus is not something a computer program can do.
The most a computer program can do is building a list of cross-frequent words amongst a great amount of different texts of different nature.
For this reason, various predefined stop-word lists are proposed on several websites related to text mining and information retrieval.
Some lists are complete and well done, but others omit too many words or contain words that could have important meanings in some cases.

Here is an overview of the type of words to remove, based on the proposition from [Porter M. 2002].

- **Pronoun forms**
  - Subjects (I, you, we, …)
    - "us" is to be removed if lower-case only
  - Possessive adjectives and possessive pronouns (my, your, our, ours, yours …)
    - Many other lists included the possessive adjective "mine", which is best left out of this list because of constructions like gold-mine.
  - Reflexive (myself, ourselves…)
  - Demonstrative (this, that, these, those…)
  - Interrogative (what, who, whom, …)
- **Verbs**
  - Verb forms of to be (am, is, are, being, been…)
  - Verb forms of to have (had, having, has, …)
  - Verb forms of to do (does, did, doing, …)
- **Auxiliaries**
  - Will, shall, can, may, must, ought and derived forms like should, could, …
  - Following homonyms however are to be discussed
    - *he made a WILL, old tin CAN, merry month of MAY, a smell of MUST*
- **Articles**
  - A, an, the, …
- **Others (Figure 1.2)**
  - prepositions, conjunctions, adverbs, …

And	again	about	each	down
But	further	against	few	in
If	then	between	more	out
Or	once	into	most	on
Because	here	through	other	off
As	there	during	some	over
Until	when	before	such	under
While	where	after	no	so
Of	why	above	nor	than
At	how	below	not	too
By	all	to	only	very
For	any	from	own	
With	both	up	same	

Figure 1.2 : other stop-words

- **Some examples of common English words (Figure 1.3)**
  - May be deleted due to their commonness
  - Once more, this all depends on the context. If the texts speak about sports, "put" will not be included in the stop-word list because a "put" is an attempt to put a golf ball in the hole.

one	seen	another	said
every	whether	however	also
least	like	two	get
less	well	three	go
many	back	four	goes
now	even	five	just
ever	still	first	made
never	way	second	make
say	take	new	put
says	since	old	see

Figure 1.3 : commom English words

- **Various contractions**
  - I've, she'd, we'll, didn't, couldn't, daren't, etc.
  - These forms are increasingly encountered nowadays, as written English becomes less formal.

We could go a little bit further by deleting numbers (three, sixty, 1955, etc.). This is to be discussed depending on the situation and the type of the processed texts. Punctuations are also to be discussed, depending on the computations done on the texts. One possibility is to delete commas but to keep endpoints to identify different sentences.

Stop-word removal diminishes the processing time and effectiveness of text mining techniques and computations by allowing a processing of the meaningful parts of the texts only, while still keeping a copy of the original full text. Stop-word removal is a must-do step.

3.3 Stemming

Whereas stop-word removal is a purely non-ambiguous process, stemming brings up some more problems and reasoning around stemming implies an overlap of different disciplines.

Let's define the concept of stemming. Words having the same stem are usually assumed to have an equivalent meaning. For instance: *computing, computed, computes, computation* ... share the same stem; namely "*comput*".

Stemming is the procedure in which a word is transformed to its stem, based on written language and not spoken language. Usually, this implies the removal of a prefix and suffix. Note that stemming is strongly language dependant.

3.3.1 Comparison between stemming and lemmatisation

Claude de Loupy [Loupy C. 2001] identified some problems occurring when trying to bring back different words to one single concept. Firstly, words may sometimes be written in different ways. For example, Kadafi, Kaddaffi, and Khadafi point all to the same person. The solution is to bring back all those names to a single form thanks to a phonetic module which will detect that Kaddafi and Khadafi sound the same way.

Grammatical variations represent a problem and an opportunity in the same time. Some situations occur where a word has the same form as a verb, a noun or an adjective, even if they mean something completely different. For example, a plane represents a flying vehicle whereas "to plane" means "to aquaplane". If the grammatical type is not detected and plane is not brought back to its correct lemma, text mining techniques will find relations between accidents due to aquaplaning and the new plane designed by Airbus. Knowing the grammatical type of a word brings up two things.

Firstly, detecting the grammatical type of a particular word makes lemmatization possible, which brings a word back to its normalized form instead of its stem. Loupy has tested the effectiveness of stemming and lemmatisation on the precision (fraction of retrieved documents that are relevant) and recall (fraction of relevant documents retrieved) values of

information retrieval systems. He concludes that stemming achieves much better results, but that the best results are achieved when both techniques are applied together. The main reasons of the difference in performance may be explained by the fact that in a lemmatisation process, the system has to deal with unknown words, which he will not be able to bring back to their lemma. This can have an impact on the performance of a solution. [Loupy C. 2001]

Secondly, detecting the grammatical type of a particular word eventually allows automatic stop-word removal. Words that are identified as articles may be deleted. Even if it is a much more complex method than the classical method with a stop word list, automatic stop-word removal has an advantage over manual stop-word removal for special cases like "do", which is sometimes a musical note, not always an auxiliary verb that we might delete.

3.3.2 Stemming in different languages

Let's analyse some particularities of different languages concerning stemming. The principle is quite simple for Indo-European languages, as Porter himself states:
Assuming words are written left to right, the stem, or root of a word is on the left, and zero or more suffixes may be added on the right. If the root is modified by this process it will normally be at its right hand end. And also prefixes may be added on the left. So unhappiness has a prefix ***un****, a suffix* ***ness****, and the* ***y*** *of happy has become* ***i*** *with the addition of the suffix. Usually, prefixes alter meaning radically, so they are best left in place (German and Dutch* ***ge*** *is an exception here). But suffixes can, in certain circumstances, be removed. So for example happy and happiness have closely related meanings, and we may wish to stem both forms to happy, or happi.* [Porter M. 2001]

Not all languages can benefit from stemming. It is applicable to Indo-European languages and Uralic languages but it is not applicable to Chinese for example. Each language has its subtleties. Just as an example, the French language possesses some classical problems; namely the fact that a particular stem can sometimes be formed from words with totally different meanings, as pointed out by Claude de Loupy. An example is "port", which can come from the verb "porter" (to carry) or from the noun "port" (harbour). Another problem for the French language is that some words have very complicated stems. A simple stemming algorithm will not be able to bring the complex word back to a valid stem. An example of such a complex word is "yeux", whose singular form is "oeil". The suffix structure of English

lies in fact mid-way between the Germanic and Romance groups, and it therefore requires separate treatment.

3.3.3 Porter's algorithm

Porter's algorithm is a well-known stemming algorithm, publicly available in different programming languages and for different languages. If we compare the most used stemming algorithms, Porter is the quickest of all. The algorithm plays with three basic categories of suffixes. There is a point that has to be underlined when talking about the Porter's algorithm. Porter himself draws attention to the fact that many implementations of the algorithm all over the web are faulty with this remark:

Researchers frequently pick up faulty versions of the stemmer and report that they have applied 'Porter stemming', with the result that their experiments are not quite repeatable. Researchers who work on stemming will sometimes give incorrect examples of the behaviour of the Porter stemmer in their published works. [Porter M. 2001]

The classical example on which a stemming algorithm can be tested is the word "agreement", which should be stemmed to "agreement". It should not change at all, it should not become "agreem" or "agree".

3.4 Bag of word representation

After having applied stop-word removal and stemming, we have to find a way to represent the text in a practical way in order to perform computations more easily and to score some words within the document.

In the bag-of-word representation, each document is represented as a vector counting the number of occurrences of the different words, which makes computations much easier.

A document $d \in D$ is represented as $\vec{t}_d = (tf(d,t_1),...,tf(d,t_m))$

Where $tf(d,t_1)$ = number of occurrences of term t_1 in document d

This is in fact similar to the Vector Space Model proposed by [Salton G. 1989].

Figure 1.4 shows the result after having performed the different steps on the following example:

d_1 = "This is a sample text for a thesis about text mining, which is related to data mining."

Occurrences	Stem	
1	sampl	t_1
2	text	
1	thes	...
2	min	
1	relat	
1	data	t_6

Figure 1.4 : bag of word represenation

When a collection of documents possess a clear predefined structure, this classical bag-of-word representation is not necessarily a good choice because the importance of the words also depends on the section in which they appear.

Documents handled by our chatterbot described in chapter 3 fortunately possess two important features.

Firstly, the documents are chosen for their nature and structure. Documents with many sections are avoided.

Secondly, the documents have been handled for structure within the web mining component beforehand.

3.5 Summary of the preparation phase

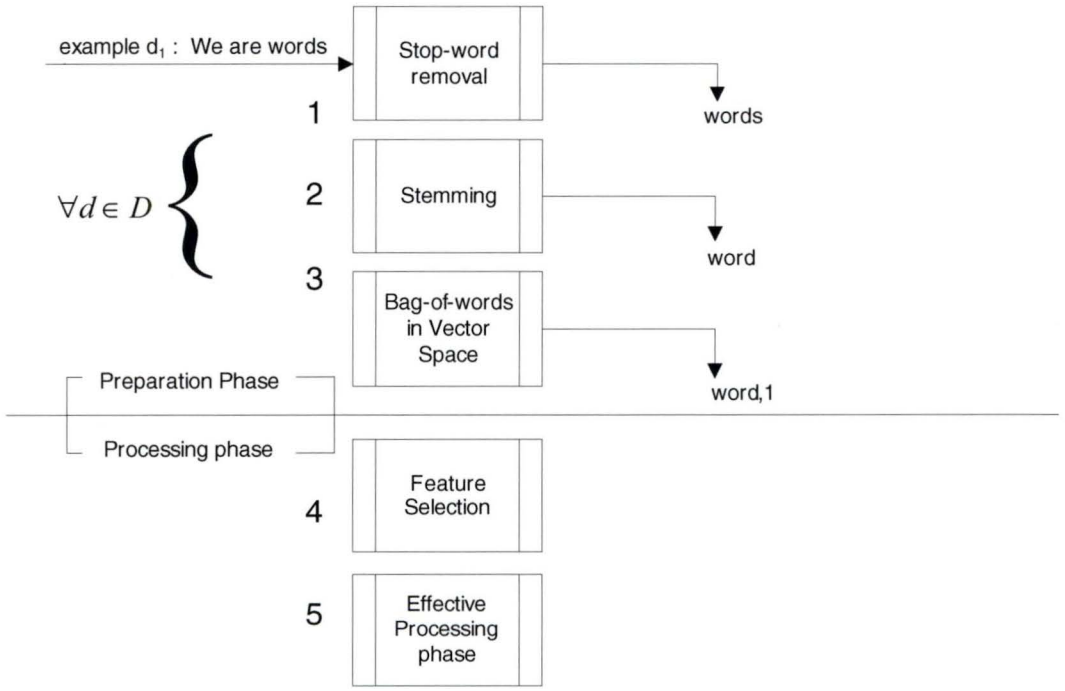


Figure 1.5 : preparation phase

Figure 1.5 represents our analysis of the preparation phase of a text mining module such as our chatterbot's. Some thoughts about the processing phase are given in the next section. Chapter 3 will cover the implemented features.

4. Processing phase

4.1 Introduction

This section briefly describes some important text mining techniques used nowadays. The techniques described in this section were chosen for their likelihood to be useful to a chatterbot. At the end of each technique, we will present a reflection of the usefulness in the case of a chatterbot.

4.2 Feature Selection or keyword extraction

Here we get to the heart of the matter. Feature selection is either a step to do before applying another text mining algorithm, or it can be the final aim on itself.
As it is easy to notice, advantages for a reader in front of an information overload problem can be solved by keyword extraction, which makes browsing trough a large text database easier by allowing the database to be indexed by the keywords of the different texts.

TFIDF

Other formulas exist throughout the literature, but TFIDF is the most widely used off all. Using term frequency (tf) and inverse document frequency (idf), it is possible to assign weights to each term in a document to represent the term's level of importance. Term frequency measures how often a particular term occurs in the entire collection. Inverse document frequency indicates the specificity of the term and allows terms to acquire different strengths or levels of importance based on their specificity.
Its aim is to score individual words within text documents in order to select concepts that accurately represent the content of the article. Classification and clustering for example can be done on the documents as soon as representative concepts have been found.

It is again [Salton G. 1988] that thought about TFIDF concept weights.

$$tfidf(d,t) = \log(tf(d,t)+1) * \log\left(\frac{|D|}{df(t)}\right)$$

Where:

D is a collection of documents and $d \in D$

$tf(d,t)$ is the frequency of term t in d

$df(t)$ represents the invert frequency of t across D *(in how many documents does t appear)*

What this formula exactly means is that a word appearing 3 times in 20 different documents will be a bad choice for clustering and classification whereas a word appearing 15 times in only 3 different documents will be a much better choice.

TF

Some text mining applications only use the plain $tf(d,t_1)$ formula.

OTHER

An optional pre-processing step prunes terms when $\sum_{d \in D} tf(d,t) > \delta$

4.3 Similarity measures

Widely used similarity measures between documents include the counting of the number of common words, the cosine angle and Euclidean distance.

Counting the number of common words

This measure is only effective if great care has been taken in the preparation phase and if additional word disambiguation has been performed.

The cosine measure

In the vector space model, the cosine measure is one of the most used similarity measures between documents. It is defined by the cosine of the angle between two vectors:

$$\text{similarity}\ (d_1, d_2) = \left(\frac{d_1 \bullet d_2}{\|d_1\| \cdot \|d_2\|} \right)$$

where • denotes the vector dot product and $\| \ \|$ denotes the length of a vector

Euclidean distance

The Euclidean distance between two vectors is represented as follows:

$$|d_1 - d_2| = \sqrt{\sum_{i=1}^{n} (d_{1i} - d_{2i})^2}$$

We have to note that the cosine measure and the Euclidean distance are not restricted to evaluate similarity between documents represented in the vector space model; they may also be used for measuring similarity between terms represented in the vector space model in the form of a word-by-word co-occurrence matrix instead of a document-by-word matrix.

Mean of a set of vectors

For some text mining methods, we will need to be able to determine the mean of a set of vectors, represented by $\vec{\mu} = \frac{1}{|D|} \sum_{d \in D} d$

For a chatterbot

As we will see in the next sections describing the different techniques, similarity measures are usually just parts of other techniques. Therefore we can already say that similarity measures are certainly helpful for every chatterbot using text mining techniques.

4.4 Keyword extraction from a single document

As seen in the pre-processing phase, the popular tfidf measure extracts keywords that appear frequently in a document but don't appear frequently in the rest of the corpus. Keyword extraction from a single document is usually an aim on itself instead of a pre-processing step.

Highlighting keywords in a single document has many applications that the tfidf measure cannot offer. Tfidf needs a large corpus to be effective, which keyword extraction from a single document does not need.

Even if the technique doesn't really summarize a text (summarization will be discussed later), it is quite useful to extract the main keywords from a paper or webpage.

According to Hearst, word count is sometimes sufficient for document overview, but in most cases, it is preferable to have a more powerful tool.

[Matsuo Y., Ishizuka M. 2003] proposed a keyword extraction algorithm based solely on a single document. We have decided to describe because the solution could certainly be applied to a chatterbot.

The conclusion of Matsuo's and Ishizuka's paper is quite impressive, because performance is nearly as good as tfidf whereas a huge corpus is not needed to achieve those good results.

Note that stemming and stop-word removal are of course must-do steps.

First, frequent terms are extracted. Co-occurrences of a term and frequent terms are counted. If a term appears selectively with a particular subset of frequent terms, the term is likely to have an important meaning. The degree of bias of the co-occurrence distribution is measured by the χ2-measure. [Matsuo Y., Ishizuka M. 2003]

In a single document, the method is logically based on sentences.

Two terms in a sentence are considered to co-occur once. If we assume that N represents the number of different terms in the document, the method computes a $N \times N$ co-occurrence matrix. Beforehand, a table of the most frequent words has been computed, we will denote it as G.

In order to evaluate statistical significance of the biases, the χ^2 test is proposed which is very common for evaluating biases between expected frequencies and observed frequencies. For each term, frequency of co-occurrence with the frequent terms is regarded as a sample value;

a null hypothesis is that "occurrence of frequent terms G is independent from occurrence of term w," which we expect to reject.

We denote the unconditional probability of a frequent term $g \in G$ as the expected probability p_g and the total number of co-occurrence of term w and frequent terms G as n_w.

Frequency of co-occurrence of term w and term g is written as $freq(w, g)$

$$\chi^2(w) = \sum_{g \in G} \frac{(freq(w,g) - n_w p_g)^2}{n_w p_g}$$

Terms with a large χ^2 value are relatively important in the document; terms with a small χ^2 are relatively trivial. We will not go too deep, but the method can be further improved by considering the length of the sentences, because a word will of course co-occur more often if it is present in long sentences.

As a final result, we obtain the following algorithm:

1) Pre-processing (stemming and stop-word removal)
2) Selection of the most frequent terms
3) Clustering frequent terms to obtain clusters denoted as C *(cf. sections 4.5 and 4.6)*
4) Calculation of expected probability $p_c = n_c / N_{total}$, where n_c is the number of terms co-occurring with $c \in C$
5) Calculation of χ^2 values. For each term w, count co-occurrence frequency with $c \in C$, denoted as $freq(w,c)$. Count the total number of terms in the sentences including w, denoted as n_w
6) Output keywords by choosing those with the highest χ^2 values

For the chatterbot

For a chatterbot, this method would be a great choice, because it doesn't need a big corpus of texts to be retrieved in order to function correctly, and it doesn't need complex computations across different texts. Performance doesn't decrease when the amount of knowledge (number of texts) increases.

4.5 Document Clustering

Let's give a general definition of what clustering is: Clustering is a grouping of a number of similar things; "a bunch of trees"; "a cluster of admirers"

Given a source of textual documents, applying the definition above, replacing *things* with *textual documents.* In order to group those textual documents efficiently, we need a similarity measure as we have seen above. Once the similarity measure has been chosen, we can find several clusters of documents that are relevant to each other according to the chosen measure.
We generate collections with the following properties:

- Documents in one cluster are more similar to one another
- Documents in separate clusters are less similar to one another

The goal is of course to achieve a high intra-class similarity associated with a low inter-class similarity. Clustering methods are able to discover hidden patterns within textual documents.

After the dimensionality has been reduced with stemming, stop-word removal and tfidf, and once a similarity measure has been chosen, document clustering may basically benefit from the same methods as traditional data clustering.

Let's discuss the 2 different approaches of document clustering (based on the data mining course given by Professor Monique Noirhomme - FUNDP).

4.5.1 Partitioning methods:

A good example is the k-means algorithm, implemented in 4 steps that construct a partition of n documents into a set of k clusters.

1. Partition objects into k nonempty subsets.
2. Compute points as the centroids of the clusters of the current partition.
3. Assign each object to the cluster with the nearest point.
4. Go back to Step 2, stop when no more new assignment take place.

4.5.2 Hierarchical Methods:

As opposed to partitioning methods, hierarchical methods don't ask for a fixed number of clusters beforehand. Agglomerative hierarchical methods are the most used for text clustering. Each document starts in a single cluster. There is a grouping of the closest pair of clusters at every iteration. This grouping depends of course on the inter-document similarity that has been chosen before performing the effective algorithm.

Divisive hierarchical methods (as opposed to Agglomerative) start with all documents in one single cluster and iterations divide them in separate clusters.

Hierarchical methods need a termination condition to avoid the end-state of both agglomerative and divisive iterative methods, which respectively bring us one single cluster or one document per cluster.

For the chatterbot

The ideas behind document clustering can be useful when it comes to choose an appropriate text (in the sense of similar) in relation with another text. Unfortunately, in a chatroom, only short phrases are exchanged. The idea of pure document clustering (coupled with summarization that we will see later) is to retain in case of a future evolution to a forum-chatterbot, where interventions are long enough to allow this technique to be useful.

4.6 Term Clustering

The basic idea behind term clustering is to work at the word level to develop models usable in indexing for other document level tasks, like automatic thesauruses.

A thesaurus is a dictionary of synonyms, but the concept has evolved to include semantic relations between concepts.

A hypothesis is that similar terms occur in similar contexts.

Basically, the idea uses the cosine similarity function at a term level and applies a classical clustering algorithm that can be hierarchical or not.

Some other well-known approaches are based on co-occurrence; namely similarity-based clustering and pairwise clustering.

These methods are more used to capture the main semantic dimensions in a text collection, avoiding synonymy and polysemy problems. I can be seen as an effective dimensionality reduction method.

Similarity-based clustering

If terms w1 and w2 have similar distribution of co-occurrence with other terms, w1 and w2 are considered to be the same cluster.

Pairwise clustering

If terms w1 and w2 co-occur frequently, w1 and w2 are considered to be the same cluster.

	a	b	c	d	e	f	g	h	i	j	...
c	**26**	**5**	—	**4**	*23*	**7**	**0**	**2**	**0**	**0**	...
e	**18**	**6**	*23*	**3**	—	**7**	**1**	**2**	**1**	**0**	...

Figure 1.6 : example of term clustering

Figure 1.6 shows an example of two (transposed) columns extracted from a co-occurrence matrix. Similarity-based clustering centers upon boldface figures, and pairwise clustering focuses on italic figures [Matsuo Y., Ishizuka M. 2003].

4.7 Classification

Let's give a general definition of what classification is. Classification is the act of distributing things into classes or categories of the same type

In fact, given a collection of labeled records (training set), each record containing a set of features (attributes), and the true class (label), the purpose of classification is to assign previously unseen records to one of the predefined classes, as accurately as possible.

A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with the training set used to build the model and the test set used to validate it. The model for a class is a function of the values of the features of the texts.

We won't detail the different methods that can be applied to obtain text classifications, but it is nevertheless interesting to mention the common techniques:

- Instance-Based Methods
    - Like k-nearest neighbors well known from classical datamining
    - Texts can be represented in Euclidean space thanks to its features
- Decision trees
    - Needs a specialized type of texts and a specific attribute selection algorithm
- Bayesian classification
    - The classification problem may be formalized using probabilities
- Neural networks

Difference between Classification and Clustering

The difference lies in the fact that there are no predetermined classes or groups when we speak about clustering.

Classification can be seen as supervised learning, because the training data are accompanied by labels indicating their class, and new data is simply classified based on the training set.

Clustering can be seen as unsupervised learning, because the class labels are unknown, the aim of the measurements is to establish the existence of classes or clusters in the data.

For the chatterbot

We don't need classification for a chatterbot, simply because of the fact that we don't need the texts to be put in predefined classes.
The texts that our chatterbot retrieves for instance (cf. chapter 3), are all of the same type, namely news articles. Eventually, it is helpful to divide the news articles into main categories (war, financial, politics, natural phenomenon etc.). A method to do it is developed in point 4.5 of the second chapter. But two reasons make this not necessary in all the cases.
Firstly, it is easier to retrieve the news articles from a WWW source that already classified the news articles. It is what we did in our application of a chatterbot which fetches financial news articles from another source as global world news.
Secondly, the answer given by the chatterbot doesn't necessary have to be based on a particular predefined category, but it simply has to be linked in any way with the last intervention on the chatroom.

4.8 Summarization

Automated summarization uses the most relevant words, sentences or extracts found in a text. It sometimes goes even further by building new phrases, not present in the original text. Automated summarization is one of the most complex fields in text mining, and it is technically very long to explain.
This section will give a description of the basic ideas behind summarization.

The goal of text summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's needs. [Mani I., Maybury M.T. 2001]

The main reason why we need automatic text summarization is that it eases information access. For example, documents can be identified faster, they may be described with keywords and they can be abstracted. The keyword extraction method that we described in a previous section is in fact keyword summarization.
In this section, we will rather analyse sentence summaries instead of keyword summaries.
Basic steps for sentence summaries include:

- Analysing the source text
- Determining its salient points
- Synthesizing an appropriate output

To summarize Hahn & Mani's ideas, we will explain the way they address the importance of a sentence. [Mani I., Hahn U. 2000]. A linear model of salience is often applied using a set of features. The total weight of a sentence is given by the sum of those features.

The location in the text.

The title, the sentences in the beginning of a paragraph, etc. will be likely to contain important information.

The appearance of cue phrases

In written texts, we can usually find lexical or phrasal summaries as "in conclusion", "in this paper", etc. The approach suggests overweighting the sentences in which they occur.

Statistical significance

This is in fact the feature selection that we described in the section about pre-processing. Tfidf is the most used of all.

Additional information

For some summarization tasks, we may have additional information entered by the user like words he knows he wants to see.

For the chatterbot

At its disposal, it is interesting for a chatterbot to possesses texts that can be used in order to reply to someone in the chatroom. Of course, the texts are too long and unnatural to be sent directly on the chatroom. Summarization could therefore be interesting to use for a chatterbot, but for our chatterbot, we will see in chapter 2 and 3, that it is possible to avoid it completely with perfect results. This is due to the nature of the texts that it handles; namely news articles. For most other type of text, summarization would certainly have been necessary.

5. WordNet

5.1 Introduction

Even if most statistical and machine learning text mining techniques don't deal with deep natural language processing, there is a tool that retains the attention of anyone that has to deal with texts. We are going to speak about semantics and for a while, we will quit our lexical, statistical and purely mathematical view of texts.

Under the direction of psychology professor Miller [Miller G. 1998], the Cognitive Science Laboratory of Princeton University has developed a tool called WordNet that neatly organises the English words into synsets (sets of synonyms) and collocations(words that have a different meaning when put together), and establishes semantic relations between them.

This tool, which is called WordNet, also provides short definitions and limited information about usage.

However, it doesn't include information about etymology, pronunciation, irregular verb forms etc. like usual dictionaries. The project is being developed since 1985 and received over $3 million of funding over the years.

We'll analyse the possibilities offered by the tool for a chatterbot.

Currently released version 2.0 is the most used semantic lexicon, and its richness is astonishing.

5.2 Organisation of the terms

The database contains about 140,000 words organized in over 110,000 synsets. The words are stored in their root form only but every word can be derived to its root form. Every synset is made of a group of synonymous words or collocations. Synsets are classified in 4 main word categories; namely nouns, verbs, adjectives and adverbs. If a word has different senses, it appears in different synsets. Synsets are related to each other via different semantic relations that we analyse here. [Miller G. 1998]

- **Nouns**
    - *Synonyms*: synsets with similar meaning

- *Hypernyms*: *Y* is a hypernym of *X* if every *X* is a (kind of) *Y* →designates a whole class of specific instances.
- *Hyponyms*: *Y* is a hyponym of *X* if every *Y* is a (kind of) *X* →designates a member of a class.
- *coordinate terms*: *Y* is a coordinate term of *X* if *X* and *Y* share a hypernym
- *Holonym*: *Y* is a holonym of *X* if *X* is a part of *Y*
- *Meronym*: *Y* is a meronym of *X* if *Y* is a part of *X*

- **Verbs**
  - *Synonyms*
  - *Hypernym*: the noun *Y* is a hypernym of the verb *X* if the activity *X* is a (kind of) *Y*
  - *Coordinate terms*: those verbs sharing a common hypernym
  - *Entailment*:

    The action represented by the verb *X* entails *Y* if *X* cannot be done unless *Y* is, or has been, done.
  - *Cause:*

    The action represented by the verb *X* causes the action represented by the verb *Y*.
  - *Troponyms :* particular ways to *X*
  - *Sentence frames:* ways the word may be used, acceptable sentence frames
- **Adjectives**
  - *Synonyms*
  - *Related nouns*
  - *Antonyms* : adjectives of opposite meaning
- **Adverbs**
  - *synonyms and root adjectives*
  - *antonyms* : *X* is an anthonym of Y if *X* means the opposite of *Y*
- **Verbs & Nouns**
  - *derived forms:* nouns and verbs that are related morphologically.
- **All types**
  - *polysemy count: number of synsets that contain the word*
  - *can be ordered by frequency score: how frequent the sense is*
  - *familiarity: how frequent the word is used in a particular type*

To show extended capabilities offered by WordNet, we have to detail it with an example: "Cat" is not only the purring sympathetic animal that we all know, but has also 7 other senses as a noun and 2 senses as a verb. This is given below as an example.

Noun

1. (18) cat, true cat -- (feline mammal usually having thick soft fur and being unable to roar; domestic cats; wildcats)

2. guy, cat, hombre, bozo -- (an informal term for a youth or man; "a nice guy"; "the guy's only doing it for some doll")

3. cat -- (a spiteful woman gossip; "what a cat she is!")

4. kat, khat, qat, quat, cat, Arabian tea, African tea -- (the leaves of the shrub Catha edulis which are chewed like tobacco or used to make tea; has the effect of a euphoric stimulant; "in Yemen kat is used daily by 85% of adults")

5. cat-o'-nine-tails, cat -- (a whip with nine knotted cords; "British sailors feared the cat")

6. Caterpillar, cat -- (a large vehicle that is driven by caterpillar tracks; frequently used for moving earth in construction and farm work)

7. big cat, cat -- (any of several large cats typically able to roar and living in the wild)

8. computerized tomography, computed tomography, CT, computerized axial tomography, computed axial tomography, CAT -- (a method of examining body organs by scanning them with X rays and using a computer to construct a series of cross-sectional scans along a single axis)

Verb

1. cat -- (beat with a cat-o'-nine-tails)

2. vomit, vomit up, purge, cast, sick, cat, be sick, disgorge, regorge, retch, puke, barf, spew, spue, chuck, upchuck, honk, regurgitate, throw up -- (eject the contents of the stomach through the mouth; "After drinking too much, the students vomited"; "He purged continuously"; "The patient regurgitated the food we gave him last night")

(taken from wordnet database)

[Miller G. 1998]

We see that WordNet is really impressive. It is easy to feel the potential offered by WordNet. All the different species of domestic cats are given with a description for all of them, from the most common cats like Siamese cats to rare species. We know that a cat is an

animal which in turn is something that lives on earth and has an own mind and decisional process, that it possesses bones and flesh as well as many other things expressed with the semantic relations given above. It shows how useful it can be for a chatterbot, even if it is not used to its full potential but only to apply semantic variations in a reply. The complete map of the word *"cat"* would take approximately 150 pages, and a lot of words have much more senses.

5.3 Interfacing with other softwares

When browsing all the possibilities for a particular word, we achieve a very complex net of relations. Wordnets are not meant to be represented in a flat indexed way like usual dictionaries. They are designed to be accessed by automatic text analysis applications.
The WordNet database can be used freely and connected to any software with the help of different interfaces in many programming languages. The tool can be downloaded freely and can also be browsed online via the Princeton University website.

5.4 Related projects and future of wordnets

When browsing the WWW looking for information about wordnets, we see that there is an increase of interest in the domain. Building wordnets in other languages, improving existing wordnets and sharing knowledge are key ideas for the future.

Examples of recent projects

- ItalWordNet is a large semantic database for the automatic treatment of the Italian language.
- Balkanet is a multilingual semantic network for the Balkan languages.
- EuroWordNet has produced wordnets for several European languages and linked them together.
- The Oxford English Dictionary plans to publish their own online WordNet.

Even if it is not intended to be used in this context, WordNet turns out to be a practical tool to learn new words quite quickly in English as a foreign language by allowing a visualization of semantic links with already assimilated concepts. This may be due to the fact

that the WordNet classification has been done by cognitive scientists that tried to arrange the database and the semantic links like a human brain would have done. [Miller G. 1998]

5.5 How to use WordNet in a chatterbot

Even if we didn't implement a connection with WordNet within the chatterbot described in chapter 3, we have thought about some ways to integrate it.
The two propositions that we give here are promising and worth some research on their own.

1. Using WordNet as a source of knowledge

Let us imagine that someone sends the following message in a chatroom:
"Hello Mike, I have bought a new car this morning ! "
If it appears that car is a central subject, we can say:
"Oh I like racing cars ! "
A racing car is nothing more then a hypernym for the word car. This is what we would call semantic knowledge.
Other semantic relations like holonyms, troponyms can be used in the same way. Even if a word has many senses, we have the frequency score and polysemy score, and we also have a discussion that allows to see if some words related to a sense have been used. WordNet is a future evolution that we would like to see implemented in our chatterbot.

2. As a way to use variations in a reply by use of synonyms

Thanks to a wordnet, a chatterbot can for example randomly use synonyms for a specific word instead of the word itself to add variations within a dialogue.

Chapter 2 : Web mining

1. Introduction

In chapter one, we covered automatic techniques that can be applied on texts, with the hypothesis that the texts are already available for immediate processing. This chapter will focus on the selection and retrieval of those texts.

Making the bot aware of current events is an essential condition to make him seem human. If the bot only talked only about things that happened in the past, the interlocutor would quickly notice that he speaks with a robot and immediately leave the chatroom. Especially in the context of a chatroom talk, people like to confine themselves to generalities first (for instance the latest news or the weather) before talking about more specific things. So, how can the bot find out about these current events?

We could imagine a system where people would enter news headlines into a database every day which could then be used by the bot.

Another idea is to make a partnership with a newsgroup like Reuters. The chatterbot could than directly pull the current events in a computerized format from the Reuters database and use it to speak about them. This practice is probably the easiest, but also the most expensive one.

For this purpose, we opt for an emerging research area called "web mining". *Web mining is concerned with the use of data mining techniques to automatically discover and extract information from World Wide Web documents and services* [Méndez-Torreblanca A., Montes-y-Gómez M. and López-López A. 2002].

Nowadays, the World Wide Web has turned into one of the most important source of information in the world. Every day, millions of websites are created or updated. But *with dramatically increasing numbers of sites, the problem of finding the ones of interest for a given problem gets more and more difficult* [Ester M., Kriegel H.P., and Schubert M. 2002]. Today, websites not only contain news, but also private, business, administration, and scientific information. In order to sort this problem out, web mining allows the use of automated tools in order to find, extract, filter, and evaluate the desired information and resources that we want, and in our case, the current events.

Web mining is still quite young. It is a field where a lot of things still have to be discovered and it is difficult to find information about every angle of the web mining. In this chapter, we'll describe some techniques that are very interesting in the case of a chatterbot, but there are certainly other existing techniques that work very well and are not described in this section. We have selected the best we have found.

This section is organized as follows. First we describe the related work. Then we'll give some techniques to retrieve information from the net. In a next section, we'll cover the classification of this information into a logical structure, and finally we'll conclude with a summary and a look in future work.

2. Related work

Web mining is usually categorized in three areas of interest, based on which part of the web to mine: web usage mining, web structure mining and web content mining [Kosala R. and Blockeel H. 2000]. This section gives a little summary of those three categories.

2.1 Web content mining

Web content mining describes the discovery of useful information from web contents. However, what web contents consist of could encompass very broad range data: government information, digital libraries, company databases, web applications, etc.

Basically, the Web content consists of several types of data such as textual, image, audio, video, metadata as well as hyperlinks.

Recent research on mining multi types of data is termed multimedia data mining. However this line of research still receives less attention than the research on the text or hypertext contents.

Web content data consist of unstructured data such as free texts, semi-structured data such as HTML documents, and a more structured type of data such as data in the tables or database generated HTML pages.

Web content mining is usually divided in two approaches: the agent-based approach and the database approach.

The agent-based approach involves the development of agents that can act autonomously to discover and organize Web-based information by using common data mining methods (classification, clustering, pattern matching, etc.).
The database approach is interested in the structure within Web documents. The DB view of web content mining mainly tries to model the data on the web and to integrate them so that more sophisticated queries other than keyword based search could be performed.

Web content mining has a lot of applications. *ShopBot* [Eikvil L. 1999], for instance, is an agent-based bot that compiles an overview of product offers, by extracting product information from several Web vendors and summarising the result for the user. Then, for comparison-shopping, the extracted product descriptions are sorted by price. *STALKER* [Eikvil L. 1999], another application, is a database approach. Information on restaurants can be spread out over different Web sites. In experiments with STALKER, information is extracted from a set of Web pages where each HTML page contains exactly one restaurant review. The information extracted consists of items like the name of the restaurant, the type of food, cost, cuisine, address, phone number and the review.

In the case of this thesis, we'll need to take the database view of web mining to discover the structure of a website and to eliminate irrelevant information.

2.2 Web structure mining

Web structure mining tries to discover the model underlying the link structure of the Web. In the database view of web content mining we are interested in the structure within web documents (intra-document structure). In the web structure mining we are interested in the structure of the hyperlinks within the web itself (inter-document structure). The model is based on the topology of the hyperlinks with or without the description of the links.

The way in which the different web pages are connected to each other via hyperlinks provides a new type of information. We can for instance see the web as a directed graph whose nodes are the documents and whose edges are the hyperlinks between them. The analysis of the properties of this graph gives information such as the similarity and relationship between different Web sites.

Web structure mining has a lot of applications. The *PageRank* system for instance can calculate the quality rank and relevancy of each web page of a site.
Another application is the Google.com site which uses a ranking system called "Clever" to discover authorities and hubs of web pages. *Clever* is an improvement of the HITS algorithm which will be described in section 4.4 of this chapter.

In this chapter, we'll use web structure mining notably in order to determine the borders of a site, and to classify a list of hyperlinks.

2.3 Web usage mining

Web usage mining tries to make sense of the data generated by the Web surfer's sessions or behaviours.
While the Web content and structure mining use the real or primary data from the Web, Web usage mining mines the secondary data derived from the interactions of the users with the Web. The Web usage data include the data from the Web server access logs, proxy sever logs, browser actions, cookies, user queries, bookmark data, mouse clicks and scrolls.

An application of web usage mining can be to help an organization to determine the life time value of customers, cross marketing strategies across products, and effectiveness of promotional campaigns, among other things. Analysis of server access logs and user registration data can also provide valuable information on how to better structure a Web site in order to create a more effective presence for the organization. In organizations using intranet technologies, such analysis can shed light on more effective management of workgroup communication and organizational infrastructure. Finally, for organizations that sell advertising on the World Wide Web, analyzing user access patterns helps in targeting ads to specific groups of users.

In this thesis, we won't use this kind of web mining.

3. Information retrieval

There is a tremendous amount of information available online, but much of this information is formatted to be easily read by human users, not computer applications [Lerman K., Knoblock C. and Minton S. 2001].

The documents retrieved from Web servers are HTML documents, i.e. sources with no explicit structure or schema. Each HTML document has its own structure.

We want to turn a Web source into a source that can be queried as if it were a database [Lerman K., Knoblock C. and Minton S. 2001] i.e. transforming this data into a formal representation [Méndez-Torreblanca A., Montes-y-Gómez M. and López-López A. 2002]. It is done by what is called a web crawler or a wrapper.

3.1 Static/manual crawler

A first possibility to extract information from a web source is to analyse the structure of this specific web page and find out its layout and format. When this is done, it is easy to build a template for this page and extracts the information that we want.

For instance the page http://www.abc.net.au/news/australia/weather/ taken from the ABC.net site contains a 4 day weather forecast with a very short comment for each day. The weather is available for most Australian cities. Figure 2.1 shows a print screen of this page.

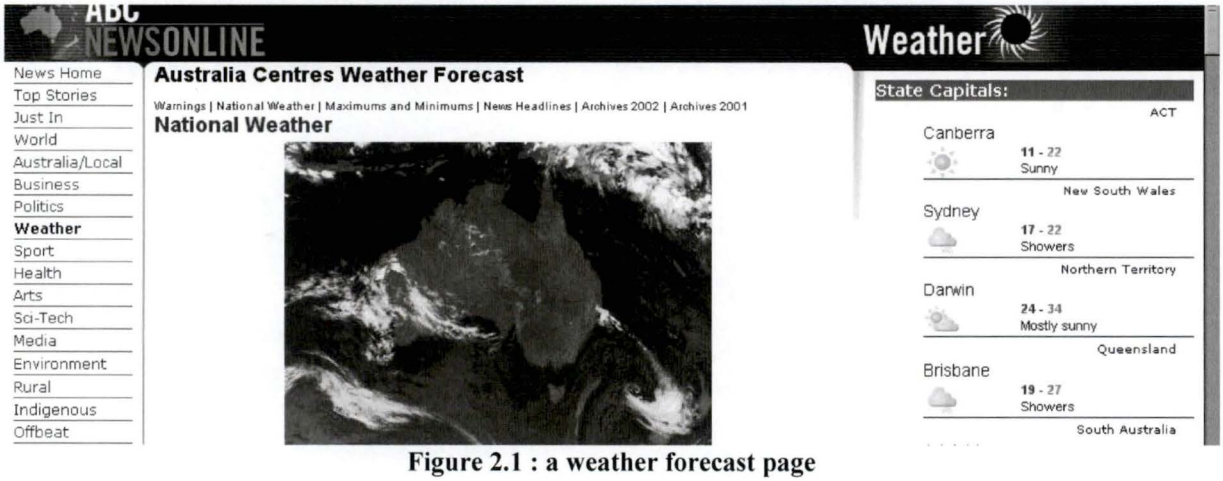


Figure 2.1 : a weather forecast page

If we make a zoom of the right part of Figure 2.1, we see that for each city, the same structure is followed: first the name of the city, then a little picture which shows a sun or clouds, then two numbers which indicate the maximum and the minimum temperature of the

day, then one or two words explaining the weather in words, and finally the state of the city. This is illustrated on Figure 2.2.

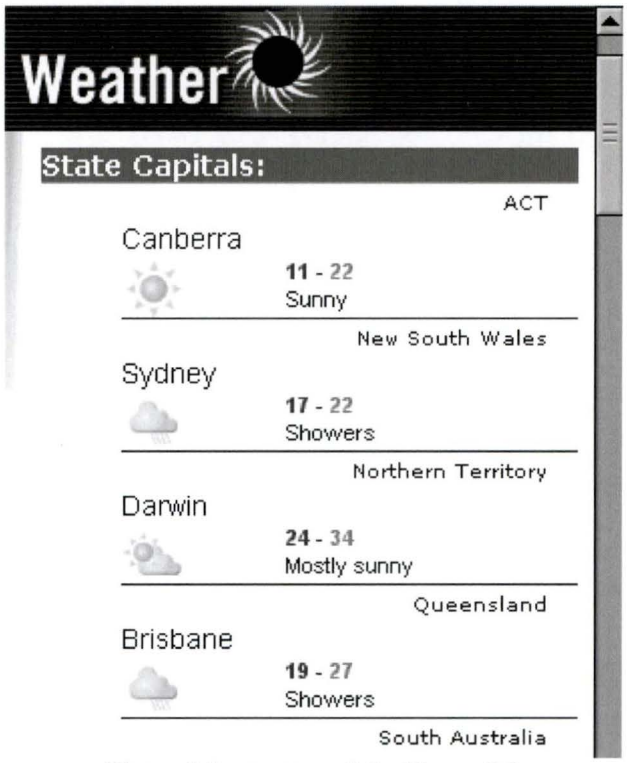


Figure 2.2 : a zoom of the Figure 2.1

It's possible to build a template whitch looks every day into the HTML code and takes the temperatures and description for each city automatically as long as the structure of the webpage doesn't change.

The templates are hand-written, so we can be sure that the extracted information corresponds to what is needed. But the creation of the templates (understand the structure of the document, writing the ad hoc code) takes a lot of time.
Moreover, the template will stop working as soon as the layout or the address changes, so these kinds of crawlers require high maintenance costs.

This approach focuses on a one-time analysis of web sites and cannot deal with constantly changing web sites, such as news sites where the information is constantly added or modified [Méndez-Torreblanca A., Montes-y-Gómez M. and López-López A. 2002].

This kind of web crawler is useful for very specific information available on a small amount of good structured and often updated web pages.

On Figure 2.1 we see that it wouldn't be easy for an automatic crawler to find out the weather for the next day in a specific town. There is a lot of other information on the page and the temperature only takes a small part of the page.

In conclusion we can say that for some specific current events (weather, lotto, horoscope ...) a static version of a web crawler is enough. For more general events (sports, politics, economics...) a more generic version has to be found.

3.2 Dynamic/generic crawler

If we have to extract information from a bigger amount of web pages, it is impossible to develop a template for each source.

The idea is to start with a list of main urls which are a selection of the best sites containing current events. The web crawler operates as follows (inspired by [Méndez-Torreblanca A., Montes-y-Gómez M. and López-López A. 2002]):

1. Resource finding: making a list of web resources to start with.
2. Downloads the page from the first URL of the list.
3. Analyse its hyperlinks: add them to the queue if the URL isn't yet in this queue, wasn't previously downloaded (in order to avoid circles), and doesn't belong to another web site (advertisings). This will be seen in section 4.
4. Stores the content of the page for its classification. This will be seen in section 5.
5. Repeats the steps 2 to 4 until the queue of URLs is empty. This condition means that the web site was fully explored.

3.3 Resource finding

By resource finding, we mean the manual selection of web resources. We have to start somewhere. In our case, we have to find current events, so the more logical start is a list of news group sites. At present, the biggest news groups are CNN, BBC, Reuters, CBS, FOX, NBC, and ABC. All of them have an official homepage. The Google News website is also very interesting. Figure 2.3 shows a print screen of the ABC web page. It's one of the most visited websites in the world and contains continuously updated information.

Figure 2.3 : The ABC web page

Figure 2.3 shows a print screen of the FOX web page. Like the ABC site, it has continuously updated information. On these two examples (which are representative of all the others quoted above), we see yet that there are some redundancies: the presence of more advertisings than content and some identical menu links (weather, politics, videos, …). We'll see the importance of those redundancies in the continuation of this chapter.

Figure 2.4 : The FOX web page

This first step, the selection of web resources, is essential for the good development of the following steps. It is important to select web sites which are often updated and contain informative and correct information. It's only with good information sources that we'll be able to extract good information.

3.4 Download a page

The downloading of a page is quite easy. For instance the URL class from java allows a simple download.

The java fonction "URL" with some manipulations gives the HTML code of a certain URL. *Another implementation uses Perl and CPAN* [Xu J., Huang Y., and Madley G. 2003]. A WebCrawler can use LWP, the libwww-Perl library, to fetch a homepage. CPAN has a generic HTML parser to recognize start tags, end tags, text and documents, etc.

4. Hyperlinks Analyse

In this steps, we put all the hypertext links in a queue if the URL
- isn't already in this queue,
- wasn't previously downloaded (in order to avoid circles)
- doesn't belong to another web site (advertisings or external information)

But, how can we determine if a page belongs to another web site? We need to determine the borders of a site. Several approaches can be used:

4.1 Same domain name

The first idea is to sample only the pages located "below" the starting page [Tian Y., Huang T., and Gao W. 2003]. For example if the URL of the starting page is http:// www.bbc.co.uk, we only sample the pages sharing the base URL http:// www.bbc.co.uk. But if we analyze the links going out from this starting pages, we see that they have several bases:
- http://news.bbc.co.uk/2/hi/europe/
- http://news.bbc.co.uk/sport2/hi/default.stm
- http://www.bbc.co.uk/weather

Thus, we have at least to change this rule into sampling the pages containing the base URL bbc.co.uk, e.g. without the "www".

But we'll see that this rule isn't enough though. The FOX news site (foxnews.com) links for instance to:
- http://www.foxnews.com/other/schedule.html
- http://shop.ecompanystore.com/foxnews.com/ (advertising)
- http://foxnews.hotellocators.com/ (not advertising, but hotel booking)
- http://www.infospace.com/info.foxnws/tbar/

The second link contains the base "foxnews.com", but links to advertisings. Thus, the best technique is to sample the pages containing the base URL of the starting page, without the www, and which are in the first part of the address.

This technique gives good results, but the number of downloaded pages could be quite huge. The efficiency of web site mining crucially depends on the number of web pages

downloaded. Actually, *many classification algorithms must be performed offline, and download of a remote web page is more expensive that in-memory classification operations* [Tian Y., Huang T., and Gao W. 2003]. So, a "pruning" technique must be introduced in order to download only the most important part of a web site:

4.2 First n pages

The naive approach of reading the first n pages of a web site does not yield a good accuracy. First, the topology of a web site is a matter of individual design and therefore tends to be very heterogeneous. Many sites contain large amounts of pages providing just structure but not content. For example, animated introductions or frames are instruments of making a site more usable or attractive, but in most cases they do not contain any content recognized by a page classifier. Another important aspect is how much content is provided on a single page. One could spread the same amount of information over several pages or just use one large page containing all the information. Consequently, the total number of pages already read is not a good indicator for pruning a web site. [Ester M., Kriegel H.P., and Schubert M. 2002]

4.3 Tree pruning

Martin Ester, Hans-Peter Kriegel, and Matthias Schubert propose a "tree pruning" technique for estimating the importance of a web page and give a criterion to choose when stop downloading a website. This section is thus an application of their paper *Web Site Mining: A new way to spot Competitors, Customers and Suppliers in the World Wide Web* [Ester M., Kriegel H.P., and Schubert M. 2002]. The technique goes as follows:

In order to summarize the content of a single web page, we assign a topic out of a predefined set of topics to it. This is done by text-classification on terms using the Naive Bayes classifier. If the page doesn't belong to any of the predefined topics, we assign the *"other"*-category to it.

Ester, Kriegel, and Schubert made the observation that in many business sites, although their trades varied widely, the same categories of pages are found in most classes: company, contact, opening hours, product and services, etc. In our case, we are not interested in business sites, but in news sites. In news sites, the categories are mostly: weather, politics, videos, top stories, latest news, sports etc.

For the pruning of a new web page, we have to build a probability table for all possible transition between the topics of the news site class. To simplify, let's assume that there are only three topics in the news site class: weather (a), politics (b) and sport (c).

→	a	b	c
a	0,2	0,7	0,1
b	0,2	0,2	0,6
c	0,1	0,5	0,4
none	0,1	0,6	0,3

Figure 2.5 : the transition probabilities for the news site class

Figure 2.5 shows all the possible transitions between the topics. The values are given as example. We see that the probability to transit from the weather topic to the politic topic is 0,7. The probability to transit from the weather topic to the weather topic (staying within the same topic) is 0,2, and so on. The *none* transition indicates the probability to start with a given topic.

We call a "web site class" the class that contains all the topics of a class. We have a news web site class, a business web site class, a private homepage web site class, etc. The web site classes are noted C_i.

For all the other web site classes, a similar table must be build. It's because one of the goals of tree pruning will be to see if the web site class changes from one page to another in the page hierarchy.

To use those tables, each web site has thus to be represented by a labelled tree. It's easy to do because the structure of most sites is more hierarchic than network-like. Web pages often begin with a main directory that links to more and more specific pages. To build up the tree as the set of minimum paths from the starting page to every page in the graph. Therefore, we perform a breadth-first search through the web site graph and ignore the links to pages already visited.

After the building of the tree, the question is how to traverse the web site and where to stop following the links any further.

The pruning method is based on two heuristics about the paths from the starting page to the following pages:

- Heuristic I: The membership of a complete path in some site class is strongly depending on the topics of the pages closest to the starting page. This is the case if there is a change in the class membership of a path.
- Heuristic II: There are cases where a whole subtree and the path leading to it do not show any clear class membership at all. This is the case if the path is treated similarly by the model of any class.

To exploit these propositions, it is necessary to measure the degree of class membership for a path. For this, we use the 0-order Markov tree. The 0-order Markov tree can calculate the probability for the occurrence of a path for each class though it was trained on complete sites. This probability is $p(s|C_i)$ and yield the information about the degree that a path *s* supports a site class C_i.

$$p(s \mid C_i) = \prod_{t \in s} p(l_t \mid pre(k,t), pre(k-1,t), \ldots, pre(1,t))$$

where l = topic of a node

t = node

s = path

k = number of predecessors of a node

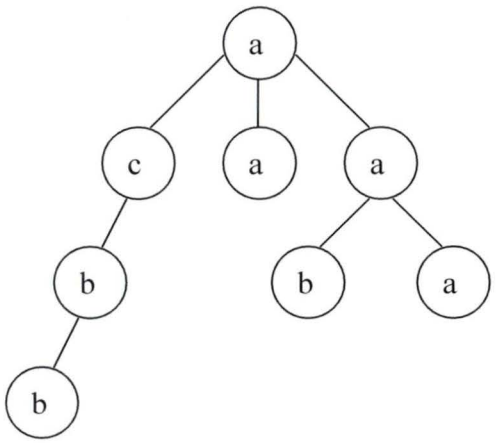
pre(k,t) = function which returns the topic of the k-th predecessor of node t.

To measure the importance of a path, we use the variance of the conditional probabilities over the set of all web site classes. The variance is a measure for the heterogeneity of the given values. A high variance of the probabilities of the web site classes indicates a high distinctive power of that particular path. Low variance means less distinctive membership.

Let *s* be any path in the site tree *S* and let $p(s|C_i)$ be the probability that *s* will be generated by the model for class C_i, then

$$\text{weight(s)} = \operatorname*{var}_{C_i \in C} \left(p(s \mid C_i)^{\frac{1}{length(s)}} \right)$$

is a measure for the importance of the path s for site classification. Let *length(s)* be the number of nodes in path *s*. The *(1/length(s))th*-power is taken to normalize *weight*(*s*) with respect to *length(s)*. This is necessary for comparing weights of paths of varying length.
Figure 2.6 shows an example of the probabilities of three paths a-c-b-b, a-c-b, and a-a-a.



→	a	b	c
a	0,2	0,7	0,1
b	0,2	0,2	0,6
c	0,1	0,5	0,4
none	0,1	0,6	0,3

p(a-c-b-b | News) = 0,1 * 0,1 * 0,5 * 0,2 = 0,001
p(a-c-b-b | News) exp 1/4 ≅ 1,18
p(a-c-b | News) = 0,1 * 0,1 * 0,5 = 0,005
p(a-c-b | News) exp 1/3 ≅ 0,17
p(a-a-a | News) = 0,1 * 0,2 * 0,2 = 0,004
p(a-a-a | News) exp 1/3 ≅ 0,16

Figure 2.6 : example of the probabilities of three paths.

To determine *weight(s)* according to the above propositions, we have to show that we can recognize a change in the class membership (heuristic I) and recognize the occurrence of unimportant paths (heuristic II). The first requirement is provided with a high probability after a certain length of the path is reached. The second requirement is obvious because a low variance means that the path is treated similarly by the model of any class.

Now a criteria for pruning the path must be applied. Due to the important role *length(s)* plays for estimating the importance of the observed path *s*, it is an essential ingredient for the pruning criterion. Let *s1*, *s2* be paths within the site tree *S*, where *s2* is an extension of *s1* by exactly one node. We stop the traversal at the last node of *s2* iff:

$$weight(s_2) < weight(s_1) \cdot \frac{length(s_2)}{\omega} \quad \text{with } \omega \in \Re^+$$

For suitable values of ω *(ω = 3)*, the criterion will very likely allow the extension of shorter paths, which should not be pruned for the following reasons. Due to the small number of nodes, the membership can be influenced strongly by the classification error of the pre-

processing step. Furthermore, our weight function can not recognize a change in the membership in very short paths. Another reason is that the question of the importance of those paths for site classification cannot be decided in such an early state.

Thus, applying the pruning rule makes no sense until descending some nodes along every path. The new figure illustrates the proposed pruning method on a small sample web site tree; in particular it shows the *weight*(*s*) values for each path *s*.

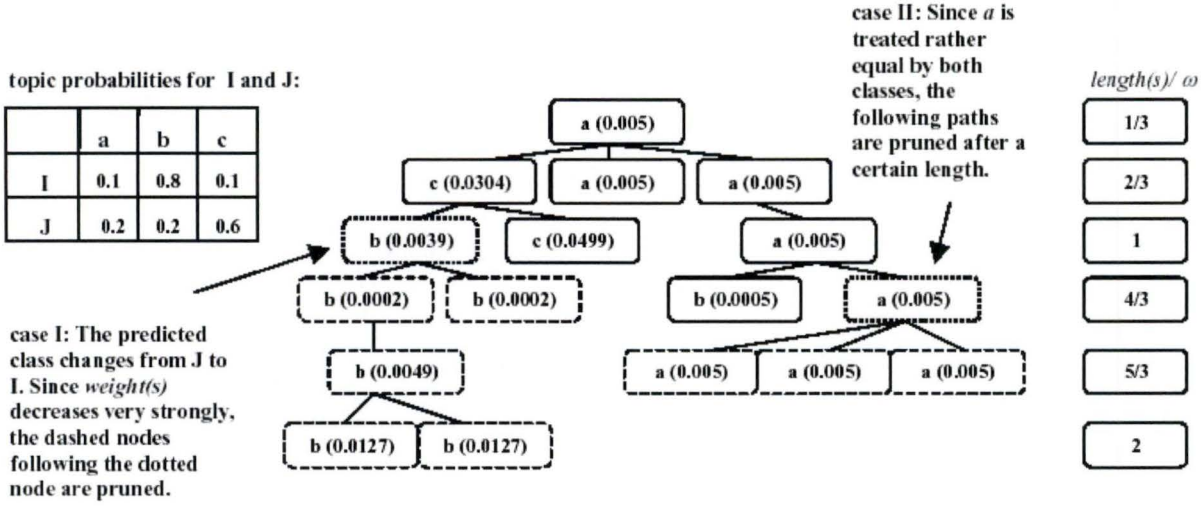


Figure 2.7 : The effect of the pruning method on the 0-markov tree classifier with ω=3.

Figure 2.7 shows a web site tree t. The nodes are labelled with the topic and the value of weight(s). The dashes nodes are pruned.

For a path *s* with *length(s)* smaller than ω, this rule will stop the traversal only if a relevant decrease in variance is observed. As mentioned above, this is interpreted as a change of the site class and we can prune any following node due to our first proposition.

For *length(s)* $\geq \omega$, the criterion is likely to prohibit the extension of a path unless a topic occurs that can significantly raise the variance. With increasing *length(s)* it is getting more and more unlikely that an additional factor can increase the variance strongly enough. Due to the required growth of variance and the decreasing influence of the additional factor, most paths are cut off after a certain length. This corresponds to the requirement made by our second proposition that a path will not provide general information about the class of the web site after a certain length.

This pruning technique allows keeping only the important pages from a complete website by cutting the unspecific subtrees.

4.4 HITS

The HITS (Hyperlink Induced Topic Search) algorithm was introduced by Jon M. Kleinberg in 1998 . It mines the link structure of the Web to evaluate the importance of links [Bar-Yossef Z. and Rajagopalan S. 2002].

Kleinberg figured out that a valuable and informative webpage is usually pointed by a large number of hyperlinks. And a webpage that points to many good web pages is itself a useful resource.

Thus, Kleinberg categorized two types of important pages: Pages with many out-bound hyperlinks and pages with many in-bound hyperlinks.

Those with many out-bound links are good directory pages and are called *hubs*, and those with many in-bound links have a good content and are called *authorities*.

The intuition is that a good authority is pointed to by many good hubs and a good hub points to many good authorities. There is also a mutually reinforcing relationship between hubs and authorities.

These two types of Web pages are extracted by iteration that consists of the following two update steps:

$$h(p) = \sum_{q,q \to p} a(q)$$

$$a(p) = \sum_{q,p \to q} h(q)$$

At the start, there is an initial assignment h(p) = a(p) = 1 for each page p. The notation p→q indicates that p links to q. Actually, h(p) and a(p) will converge to a fixed point, known as the hub and authority score respectively. The higher these scores, the more relevant are the pages.

HITS is a query-based algorithm. It works only with the query of a topic. The original algorithm works as follows:

1. Given a user query, the HITS algorithm first collects the 200 highest-ranked pages for the query from a search engine.

2. Add to the 200 pages their one-hop-away neighbours: all the pages pointed by those pages.
3. Delete all the links between pages with the same domain name.
4. Make an adjacency matrix of the link structure and normalize it to find the elements with the largest absolute values. Return them as authorities.

In our case, we would of course make a query about "news" and take the top-ranked pages in the Google search engine. If we use of the HITS algorithm, the section 3.3 isn't necessary. Google would do the resources finding for us.

HITS is thus an interesting tool to prune the links of a website by giving an authority and a hub score to each page. The pages with the highest scores are kept, and the others are pruned.

But the HITS algorithm is reported to fail in some real situations. There are several reasons for it [Bar-Yossef Z. and Rajagopalan S. 2002]:

- *HITS doesn't distinguish between links on the page. It doesn't exclude local (or nepotistic) links.*
- *HITS implicitly assumes that a page has only one topic. It doesn't point out cases where topically diverse.*
- *HITS isn't weighted by the relevance of the text surrounding it to the query. It doesn't check if the pages comes up to the lexical content of the query.*

4.5 Hyperlink Classification

In hyperlink environment, links contain high quality semantic clues to a page's topic, and such semantic information can help achieve even better accuracy than that possible with pure keyword-based classification [Tian Y., Huang T., and Gao W. 2003].

If we analyse the hyperlinks present on the main page of all the big news groups, we see that there are two types. The first type of hyperlink can be called a *general hyperlink.* Sport, business, weather, politics, Europe, science are examples of general links. The other type of link can be called a *headline link.* "Rwanda remembers genocide victims", "Another arrest over Madrid bombs", or "Haiti ex-minister held over killings" are examples of headline links.

This distinction is important for future classification of the pages and for the chatterbot. In fact, in the dialog, it isn't credible for the chatterbot to speak with long sentences. Thus, the headline links are good opportunities of sentences which can be chosen as an answer.
The global links also have an application for the chatterbot. They can be used to change the thread of the conversation, while staying in the same topic. For instance, if somebody speaks about football, the bot can answer with rugby, staying in the "sport" topic.

The question is how to distinguish between global links and headline links. Three possibilities have to be examined:

First, we see that the biggest difference between the two types of links is the number of words that it contains. A general link contains in average one or two words while a headline link contains often at least four words. The first possibility is thus to class the short links into the global links, and the longer links into the headline links.

Another possibility is to use the authorities and hubs concepts of the HITS algorithms. It seems logical that a general hyperlink links to a hub and the headline hyperlink links to an authority. An authority, by definition, is a valuable and informative webpage pointed by a large number of hubs. This is the case of the pages pointed by our headline links. They are a

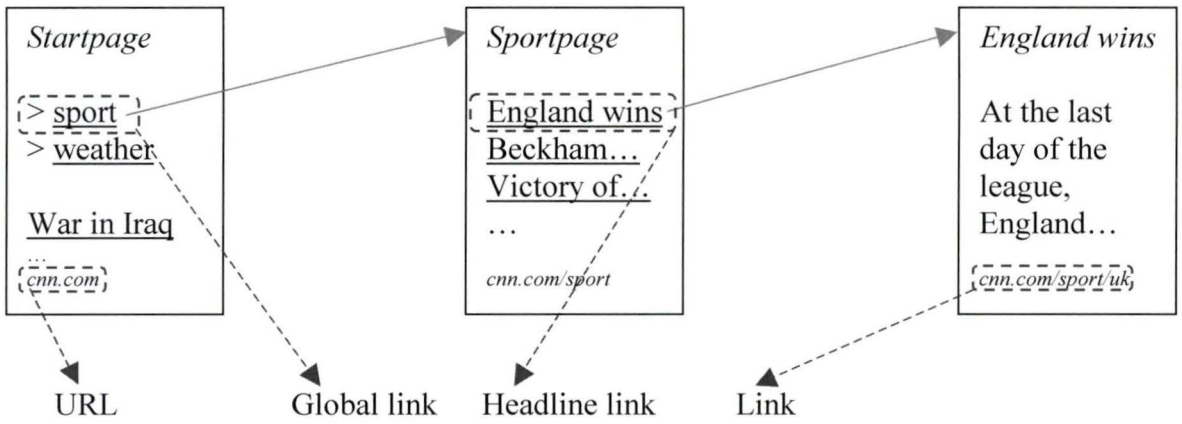
description of the headline and thus, the more a headline will be important, the more links will point on it.
A hub, by definition, is a useful directory webpage that points to many good authorities. This gives our definition of a general link. A general link usually links to a page with many hyperlinks, hyperlinks which link themselves to headlines in the topic of the general link.
Consequently, we can say that a general hyperlink usually links to a hub, and the headline hyperlink links to an authority.

A third possibility is to use the layout of the website. Global links are often in a menu of links which appears in many pages and doesn't change from one page to the other, while headline links are more in the central information block of a single page which is often updated. To determine which part of a layout belongs to the menu and which part belongs to the central information block, some template discovering techniques exist. Two of them, the noise elimination technique (point 5.2), and the hierarchical clustering (point 5.3), will be described in the future sections of this chapter.

Those three possibilities aren't exhaustive, but give a general idea of the way to classify the links into global links and headline links. Both will can be used by a chatterbot to construct its sentences.

For each link that belongs to the web site (after pruning has been applied), we keep its type, its label, and its URL into a database. Figure 2.8 shows an example with six links scattered on three pages.

The goal of this database is to have a list of all the pages (the *links* in the table) which are assumed to have an interesting content. Those pages will be exanimated with techniques covered in the next section.



Link	Type	Label	URL
cnn.com/sport	global	sport	cnn.com
cnn.com/weather	global	weather	cnn.com
cnn.com/war	headline	War in Iraq	cnn.com
cnn.com/sport/uk	headline	England wins at world cup	cnn.com/sport/
cnn.com/sport/beckham	headline	Beckham scores 3 goals	cnn.com/sport/
cnn.com/sport/victory	headline	Victory of the century	cnn.com/sport/

Figure 2.8 : example with six links scattered on three pages

The type and the label of the link are useful to classify the future content of the page. Actually, the label will be used to give a topic and a headline to a page. The headline is the label of the link if its type is "headline" and the topic is the last global link from the link chain. In the example above, we have a link chain *cnn.com* → *cnn.com/sport* → *cnn.com/sport/uk*. The headline of *cnn.com/sport/uk* is "England wins at world cup" and its topic is "sport" because the last global link in the chain is the label of *cnn.com/sport* which is "sport".

Notice that it's possible to have a link without a headline or without a topic. It's for instance the case where a headline link is taken from the start page of a website. In this case, the topic could be called "general". On the example above, the topic of *War in Iraq* is thus "global".

If we are in the case when we have a general link without a headline, which is the case of the *cnn.com/weather* link in the example above, the headline can stay empty but the page won't be chosen as an answer for the bot.

5. Content Classification

For the future needs of the chatterbot, there are two types of sentences which are important:

1) Short **headline** or introduction sentences: In the dialogue, it isn't credible for the chatterbot to speak with long sentences. Thus, the headline links are good opportunities of sentences which can be chosen as answers.
2) Long **descriptions** of the headlines: The text mining operations need long tests to "mine" the text. In this description, we can remove any non-textual information such as HTML-tags and punctuation from the documents.

Some additional information like the timestamp, the source, the link and the topic are also useful. The Timestamp, source and link are described in chapter 3. The topic can be used to change the thread of the conversation, but by staying in the same topic. For instance, if somebody speaks about football, the bot can answer with rugby, staying in the sport topic. The topic is also useful according to the theme of the chat. If it's a financial chat for instance, the bot can refocus the subject of the speech to financial things, simply by choosing a finance headline as sentence.

The final purpose of all the web mining techniques proposed in this chapter is to fill in the data base described on Figure 2.9.

Headline	Timestamp	Source	Link	Description	Topic
England wins at world cup	10716261824	ABC	abc.net.au/s101.htm	At the final of the 2003 rugby world cup…	sport
War in Iraq	10716261887	CNN	cnn.com/7693.htm	Another us army died yesterday in…	general

Figure 2.9 : the knowledge data base

To keep only the needed part of a page, we have to eliminate the noisy information and determine the actual informative content block of a page. For this goal, we'll present two techniques: Noise elimination and hierarchical clustering.

5.1 DOM

In this section, we will often use the DOM (Document Object Model) tree. The DOM tree is the representation of a HTLM page where the tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes. Figure 2.10 shows a segment of HTML codes and its corresponding DOM tree.

```
<BODY bgcolore=WHITE >
 <TABLE width=800 height=200 >
 ...
 </TABLE >
 <IMG scr="image.gif" width=800 >
 <TABLE bgcolor=RED >
 ...
 <TABLE >
<BODY >
```

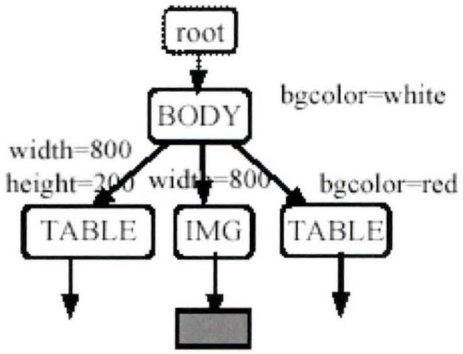


Figure 2.10 : A DOM tree example (lower level tags are omitted)

5.2 Eliminating noise with the Site Style Tree

The chatterbot needs to be aware of current events. These current events can easily be found on the above-mentioned websites (CNN, CBC, etc.), but, as introduced before, the useful information, e.g. the events on the website are often accompanied by a large amount of noise such as banner advertisements, navigation bars, copyright notices, decoration pictures, etc.

Although such information items are functionally useful for human viewers and necessary for the Web site owners, they often hamper automated information gathering and web data mining.
We call these blocks that are not the main content blocks of the page the noisy blocks.

Lan Yi, Bing Liu and Xiaoli Li [Yi L., Liu B., and Li X. 2003] propose a noise elimination technique based on the following observation:
In a given Web site, noisy blocks usually share some common contents and presentation styles, while the main content blocks of the pages are often diverse in their actual contents and/or presentation styles. Based on this observation, they propose a tree structure,

called Site Style Tree, to capture the common presentation styles and the actual contents of the pages in a given Web site. By sampling the pages of the site, the Site Style Tree (STT) can be built for the site. They then introduce an information based measure to determine which parts of the SST represent noises and which parts represent the main contents of the site. The SST is used to detect and eliminate noises in any Web page of the site by mapping this page to the SST.

Figure 2.11 shows a typical example of noise in a Web page. It's a news article of the ABC webpage. We see that the article (the dashes part) only takes a small part of the page. It's the content block of the page that is often updated. The common content of this page is the advertising banner on the top, navigation links on the left and the shortcuts in the middle.

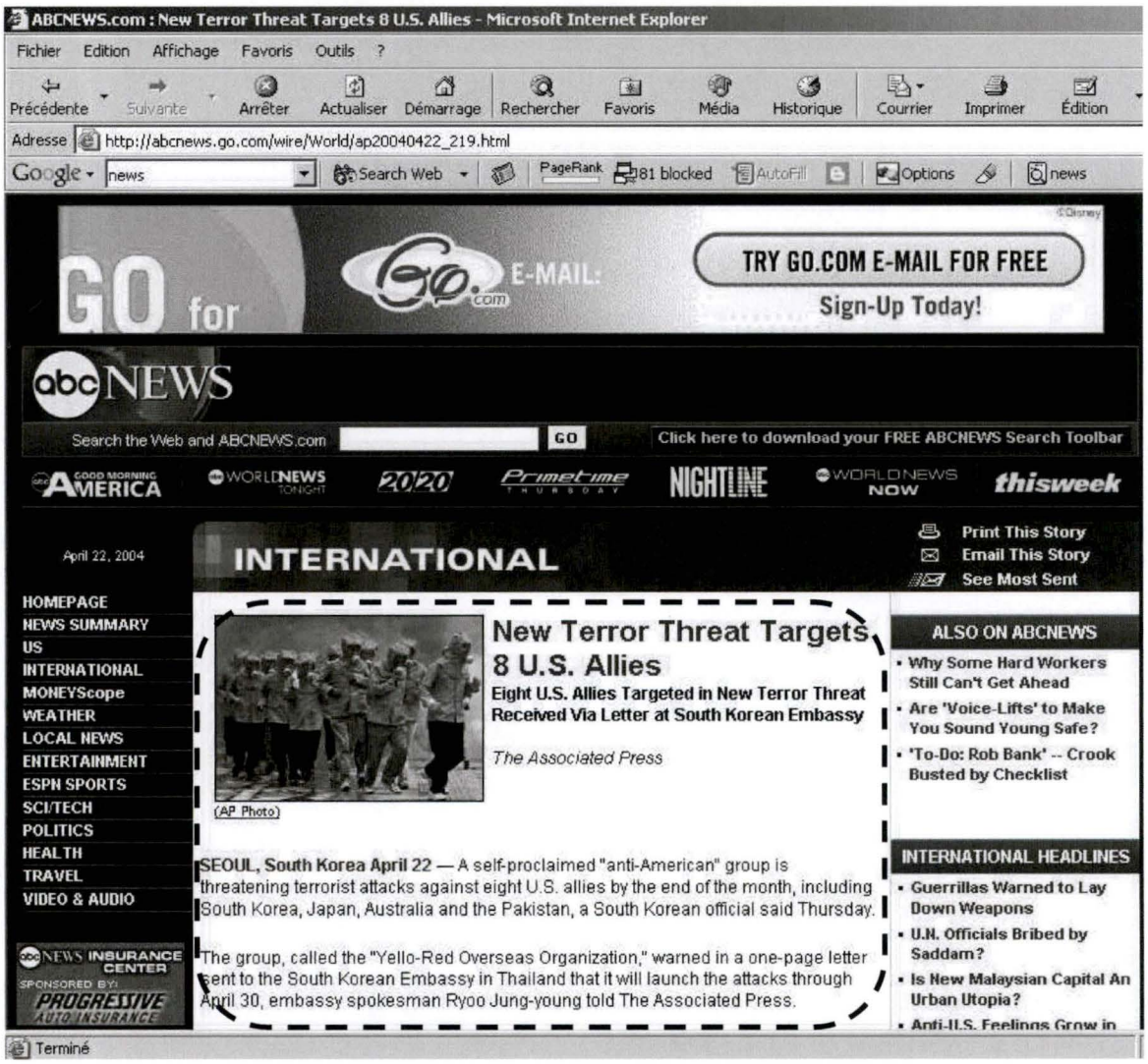


Figure 2.11 : print srceen of a news article

The proposed cleaning technique is based on the analysis of both the layouts and the actual contents (i.e., texts, images, etc.) of the Web pages in a given Web site. Thus, the first task is to find a suitable data structure to represent both the presentation styles (or layouts) and the actual contents of the Web pages in the site. Lan Yi, Bing Liu and Xiaoli Li propose a Site Style Tree (SST) for this purpose.

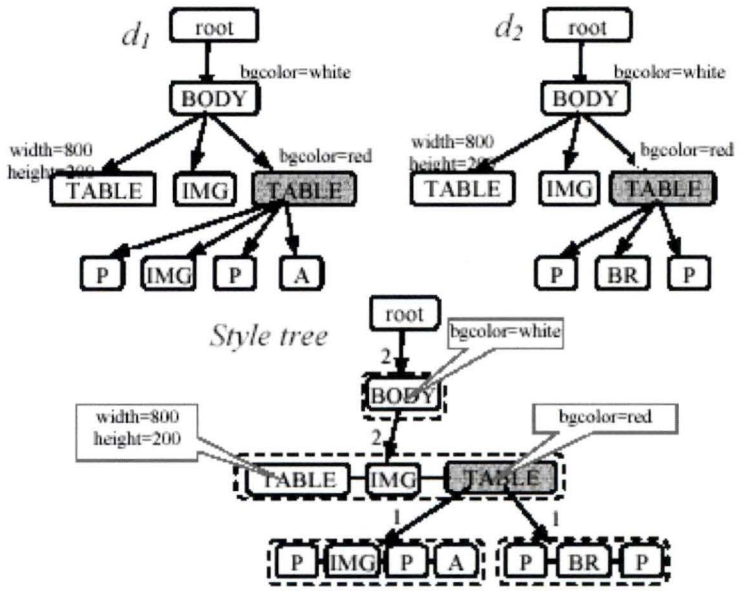


Figure 2.12 : DOM tree and style tree

A Site Style Tree is a combination of the DOM (Document Object Model) trees of a set of related Web pages. A DOM tree is commonly used for representing the structure of a single Web page. The figure above shows the combination (the Site Style Tree) of two DOM trees. A site style tree consists of two types of nodes, namely, style nodes (S) and element nodes (E).

A *style node* (*S*) represents a layout or presentation style, which has two components, denoted by (*Es*, *n*), where *Es* is a sequence of element nodes (see below), and *n* is the number of pages that has this particular style at this node level.

On figure 2.12, the style node (in a dash-lined rectangle) P-IMG-P-A has 4 element nodes, P, IMG, P and A, and $n = 1$.

An *element node E* has three components, denoted by (*TAG, Attr, Ss*), where

- *TAG* is the tag name, e.g., .TABLE. and .IMG.;
- *Attr* is the set of display attributes of *TAG*, e.g., bgcolor =RED, width = 100, etc.
- *Ss* is a set of style nodes (S) below *E*.

Note that an element node corresponds to a tag node in the DOM tree, but points to a set of child style nodes *Ss*.

To build a site style tree for the pages of a Web site, we first build a DOM tree for each page and then merge it into the STT in a top-down fashion. At a particular element node *E* in the STT, which has the corresponding tag node *T* in the DOM tree, we check whether the sequence of child tag nodes of *T* in the DOM tree is the same as the sequence of element nodes in a style node *S* below *E* (in the STT). If the answer is yes, we simply increment the page count of the style node *S*, and then go down the STT and the DOM tree to merge the rest of the nodes. If the answer is no, a new style node is created below the element node *E* in the STT. The sub-tree of the tag node *T* in the DOM tree is copied to the site style tree after being converted to style nodes and element nodes of the style tree.

Now, we have to determine the noisy elements of our STT. The definition of noise is based on two assumptions.

1. The more presentation styles that are used to compose an element node, the more important the element node is.
2. The more diverse that the actual content of an element node is, the more important the element node is.

Definition of node importance: For an element node *E* in the SST, let *m* be the number of pages containing *E* and *l* be the number of child style nodes of *E* (i.e., $l = |E.Ss|$), the *node importance* of *E*, denoted by *NodeImp*(*E*), is defined by

$$NodeImp(E) = \begin{cases} -\sum_{i=1}^{l} p_i \log_m p_i & if \quad m > 1 \\ 1 & if \quad m = 1 \end{cases}$$

where p_i is the probability that a Web page uses the *ith* style node in *E.Ss*.

For instance, in the SST of the figure below, the importance of the element node Body is 0 ($1\log_{100} 1 = 0$). That is, below Body, there is only one presentation style Table-Img-Table-Table. The importance of the double-lined Table is $-0.35log_{100}0.35 - 2*0.25log_{100}0.25-0.15log_{100}0.15 = 0.292 > 0$

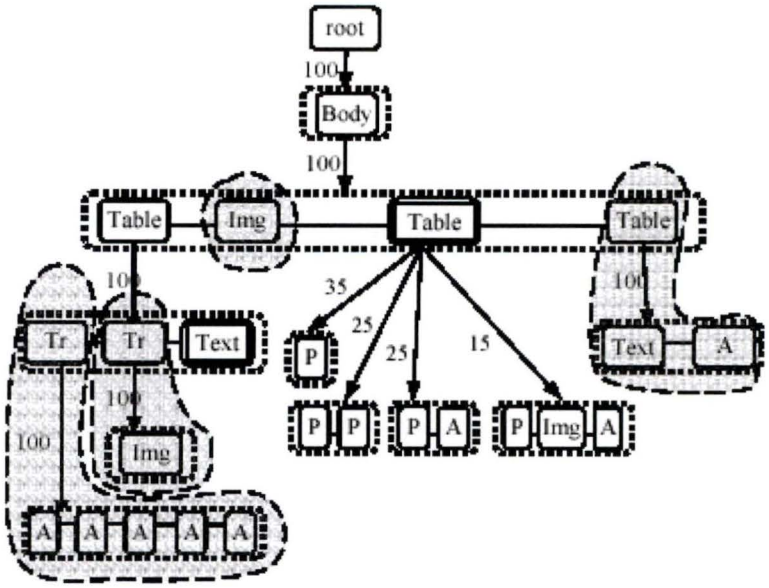


Figure 2.13 : an example of style tree

However, we cannot say that Body is a noisy item by considering only its node importance because it does not consider the importance of its descendents. Thus, the definition of the node importance has to be reconsidered to measure the importance of an element node and its descendents. This can be easily done by adding to the node importance, the average of the node importance of its descendents, multiplied by an attenuating factor. This sum is called the *composite importance* of a node.

At present, we have prioritized the presentation in the evaluation of the importance of an element node.

But the content is also important. It's why the definition of the importance of leaf nodes, such as "P" in the figure, is different from internal nodes. They have actual content with no tags. We define the importance of a leaf element node based on the information in its actual contents (i.e., texts, images, links, etc.).

Definition of composite importance: For a leaf element node E in the SST, let l be the number of features (i.e., words, image files, link references, etc) appeared in E and let m be the number of pages containing E, the composite importance of E is defined by

$$CompImp(E) = \begin{cases} \sum_{i=1}^{l} \frac{H(a_i)}{l} & if \quad m > 1 \\ 1 & if \quad m = 1 \end{cases} \quad \text{with } H(a_i) = -\sum_{j=1}^{m} p_{ij} \log_m p_{ij}$$

where a_i is an actual feature of the content in E. $H(a_i)$ is the information entropy of a_i within the context of E. And p_{ij} is the probability that a_i appears in E of page j.

The information entropy, $H(a_i)$, indicates whether the feature a_i is redundant or informative. Its value is always between 0 and 1. ($0 \leq H(a_i) \leq 1$).

Figure 2.14 is an example with m = 2. This example shows two websites with the same layout. *It's a widely used layout in dot-com web sites where the logo of a company is on the top, followed by advertisement banners or texts, navigation panels on the left, informative content on the right, and its copyright policy at the bottom* [Shian-Hua L. and Jan-Ming H. 2002].

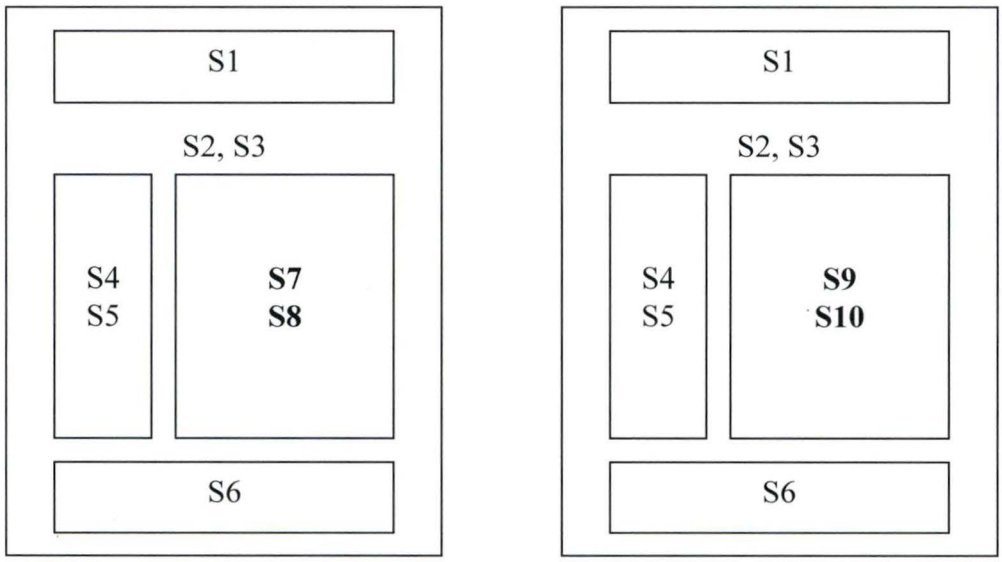


Figure 2.14 : example of the entropy

Let's make the calculation, for H(S1) and H(S7). S1 is present on both websites, and S7 is present only on the left website.

$$H(S1) = -\sum_{j=1}^{2} \frac{1}{2}\log_2 \frac{1}{2} = -\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2} = -0{,}5.(-1) - 0{,}5.(-1) = 1$$

$$H(S7) = -\frac{1}{1}\log_2 \frac{1}{1} - \frac{0}{1}\log_2 \frac{0}{1} = 0$$

We see that the closer entropy is 1, the more redundant the feature is. The composite importance of E, CompImp(E) is the 1 minus the entropy of the leaf element. It's the complementary of the average of the entropy of all its features, thus 1 minus the sum of all its features divided by the number of features.

If the value is close to 0, E is redundant, and if it's more than a defined threshold, E is informative.

We now define what we mean by noises:
Definition of noisy: For an element node *E* in the SST, if all of its descendents and itself have a composite importance that is less than a specified threshold *t*, then we say element node *E* is *noisy*.
Definition of meaningful: If an element node E in the SST does not contain any noisy descendent, we say that E is meaningful.

Notice that some element nodes in the SST may be neither noisy nor meaningful, e.g., an element node containing both noisy and meaningful descendents.
We now have all the tools to detect and eliminate noisy information from a whole website.

The first thing to do is to randomly crawl a number of Web pages from the Web site and builds the SST based on these pages. The reason of this random technique is that in many sites, we can not crawl all its pages because they are too large.

Then for each page of this website we map the DOM tree of the page to the SST built just before, and depending on where each part of the DOM tree is mapped to the SST, we can find whether the part is meaningful or noisy by checking if the corresponding element node in the SST is meaningful or noisy. If the corresponding element node is neither noisy nor meaningful, we simply go down to the lower level nodes. If it is meaningful, we keep it, and if it is noisy, we don't keep it.

At the end of the mapping, the parts of the DOM tree which are kept can be seen as the informative content of the web page, corresponding to news articles in our case.

5.3 Clustering for web information hierarchy mining

Hung-Yu Kao, Jan-Ming Ho and Ming-Syan Chen from Taiwan proposed an other information detection. This section is a summary of their paper *Clustering for web information Hierarchy Mining* [Kao H.-Y., Ho J.-M., and Chen M. S. 2003].

As in the noise elimination technique they noticed that the structure of web pages are most of the time generated dynamically by a template, and are thus similar to one another in the same web site. They noticed that the pages are usually assembled by a set of fundamental *information clusters*. They tried to make a hierarchy of those clusters, by joining the clusters with the same amount of information contained.

An *information cluster* is defined as a substructure of a page which provides a unique semantic representation to users among pages in a web site and is composed of *information elements* or smaller information clusters.

An *information element* is one context or an anchor with non-zero length.

A *clustering* is the configuration formed by the set of information clusters after the k-th level clustering, denoted by L_k.

Their information clustering system can be summarised in two steps:

1. The information coverage tree building
2. Multi-Granularity Information Clustering

We will describe these two phases with more details in the next paragraphs.

1. The information coverage tree building

In this step, we'll build an information coverage tree (ICT). Similarly to the SST of the previous section (eliminating noisy information), the ICT is the aggregated tree of several DOM trees. In this tree, each node has some calculated features to indicate the scales of the information that it contains. These features are the CII, API, and SII. We will describe them further in this section. To build the information coverage tree, we first compute the values of those features for each DOM tree of a web site, and then we aggregate them into the ICT.

The first feature is the *content information index* (CII). It's the average of the entropy values of the terms of a node. To calculate the CII, we need the entropy. As usual, the entropy of a term is calculated according to its term frequency:

$$\text{EN}(\text{term}_i) = -\sum_{j=1}^{m} p_{ij} \log_m p_{ij}$$

where m = |D|, D is the set of pages, and where p_{ij} is the value of normalized term frequency in the page set.

To get the content information index of a node N, we take the average of the entropy of the terms in an *innerText* of node N. The innerText is the context delimited by the tag and the structure of a page.

$$\text{CII(N)} = \sum_{i=1}^{k} \frac{EN(term_j)}{k}, \text{ where } \forall_{j=1\sim k} \text{term}_j \text{ in innerText of N and k is the number of terms.}$$

The value of the CII is used to indicate the scale of the *information authority* of the node. The definition of the information authority is similar to the one from HITS, but differs from it in the way that information authorities are not specific to any topic.

We also define the value of the anchor precision index (API) to indicate the correlation of the anchor and its linking page. We use the anchor text to evaluate the value of API. The correlation index API is defined as

$$\text{API(N)} = \sum_{j=1}^{m} \frac{1}{EN(term_j)}$$

Where term_j is the term concurrently appearing in both the anchor text of N and the linked page and m is the number of matched terms.

Similarly to the CII, the API is used to indicate the scale of the *information hub* of the node (i.e. if it contains information linking to information authorities).

With the CII and the API, we can express a node in the 2-dimentional space based on its scale of information authority and information hub with the tuple value (API(N)/AC(N) , CII(N)) where AC(N) is the count of anchors contained in the tree of node N. This is shown on Figure 2.15.

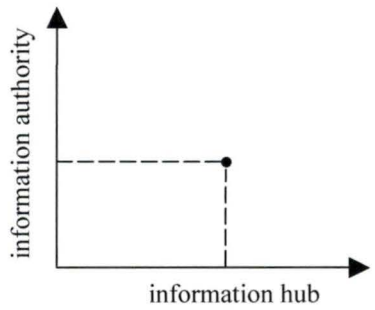


Figure 2.15 : 2-dimentional space of authority and hub

Finally, the structure information index (SII) of a node is calculated according to the distribution of the feature values of the node's children. We define the SII value of node N with children n_0, n_1, … , n_{m-1} for feature f_i as:

$$SII(N, f_i) = \sum_{j=0}^{m-1} w_{ij} \log_m w_{ij}$$

where $w_{ij} = \frac{f_i(n_j)}{\sum_{k=0\sim m-1} f_i(n_k)}$, $\forall n_i \in children(N)$ and m = number of children.

We apply entropy calculation to represent the distribution of the children's feature values of any node with more than one child. The value of SII indicates the degree that the feature values of the node are dispersed among its children. When the value of SII(N, f_i) is higher, the values of all children's f_i tend to be equal.

In our case, f_i = number of different tagging styles. It's because experts selected the pages having different and distinctive tagging styles to be the news article answer set.

Now we have found the CII, the SII and the API for each webpage. To build the information coverage tree (ICT) we aggregate those values. Each node of the ICT is called an information cluster.

2. Multi-Granularity Information Clustering

The multi-granularity information clustering is one in two steps. In the first step, we're looking for the k highest information clusters of a page. These top-k clusters will be called *information centroids*.

To choose them, we'll take the information clusters which SII values are higher than a given SII Threshold (ST). According to this ST, a top-down algorithm can mark them as information centroid. In Figure 2.16, the information clusters which have the highest values received a number.

After the top-k information centroids are marked, we apply the second step of the multi-granularity information clustering. It's a node merging process. We want to merge the neighbouring and similar information centroids and clusters into a more generalized cluster.

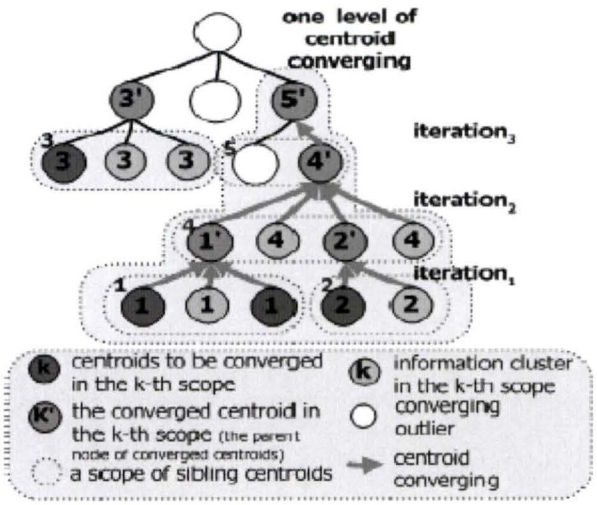


Figure 2.16 : the centroid converging in the DOM tree

We define a *scope* as a set of nodes sibling the same mother-node and containing at least one centroid. In Figure 2.16, the scopes are numbered from 1 to 5 and surrounded by a dash-line.

The converging process is quite technical. Hung-Yu Kao, Jan-Ming Ho and Ming-Syan Chen didn't explain in their paper how they discovered this method, but they say that experiments on several real news websites show very good results.

The converging process goes as follows:

For each scope of level i of the tree which contains at least one centroid :

- Verify that all the clusters have the same size of information. Geometrically, that means that all the centroids and the clusters of the scope are in a circumference of CInfoBase + i * CInfoInc. Where the tuple CInfoBase and CInfoInc are pre-assigned thresholds.
- If yes, find the new converged centroid and go one level up

Each converged centroid got a new cluster information value. It's the geometric average of the maximum difference of information authorities and the maximum differences of information hubs. If there is only one centroid in the scope, the value equals the distance between this centroid and the converged centroid.

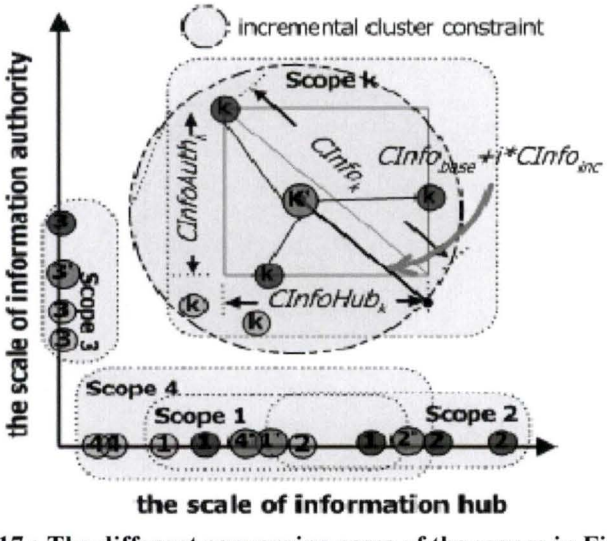


Figure 2.17 : The different converging cases of the scopes in Figure 2.16

The cluster information value $Cinfo_k$ measures the information diversity between centroids in the same scope. If this value is big, it means that this cluster contains significant information and that it has a strong likelihood to be a news article.

Figure 2.18 shows a global vue of the Multi-Granularity Information Clustering

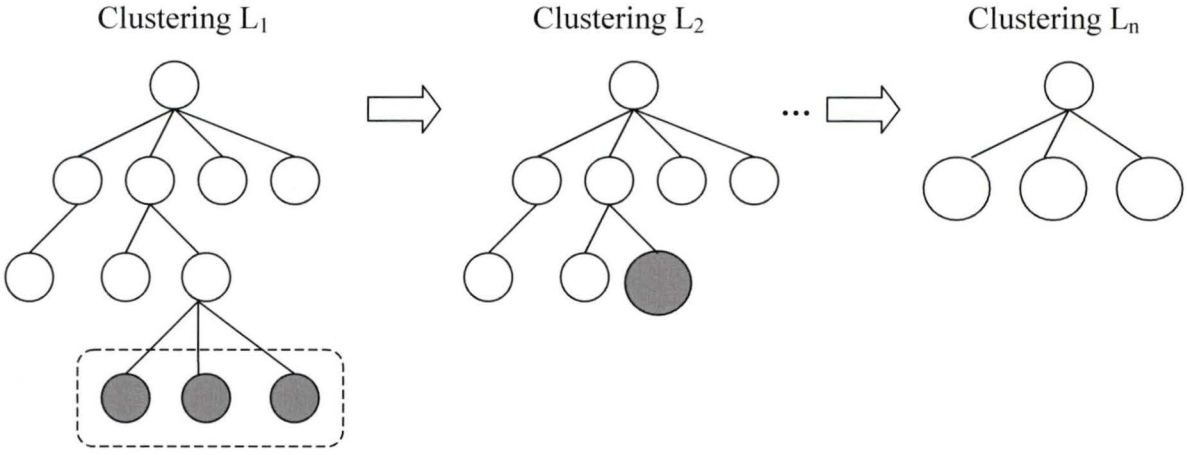


Figure 2.18 : the clustering steps

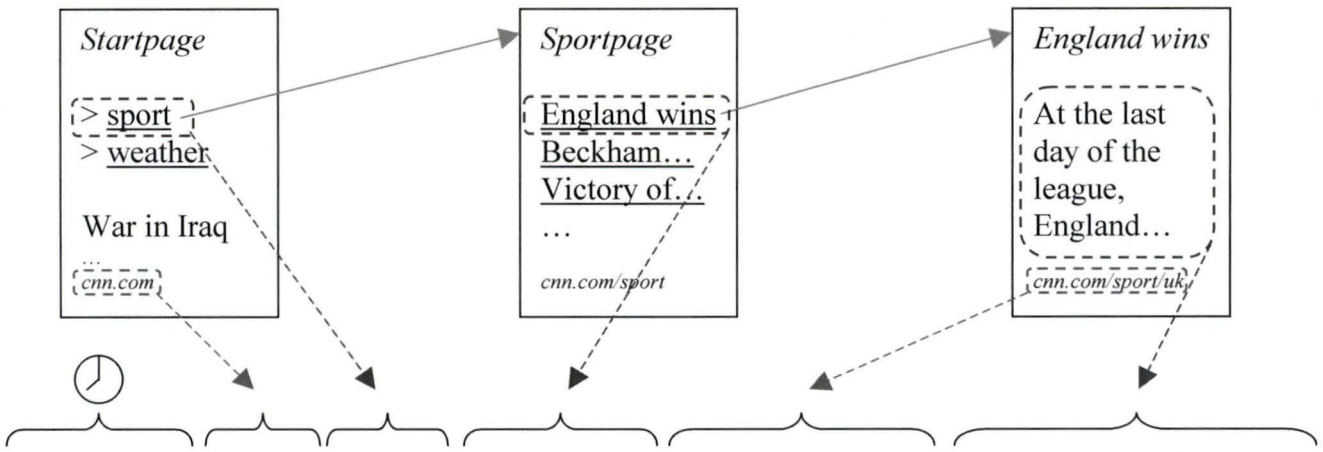
5.4 Data base insertion

The "Eliminating noise with the Site Style Tree" or the "Clustering for web information hierarchy mining" techniques allow to only keep the actual informational content from a webpage. After eliminating advertisements, menus, copyrights, etc, we are able to insert the *description* (corpus) of current news into the database.

The *headline* and the *topic* are taken from the link classification. Remember that the link classification makes a distinction between global links and headline links. The former are labelled with words like "sport", "business", "weather", "politics", etc. The latter are labelled with words like "Rwanda remembers genocide victims", "Another arrest over Madrid bombs", or "Haiti ex-minister held over killings".
We can use this link classification as explained before in such way that the headline is the label of the headline link and the topic is the last global link from its link chain.

The *timestamp* is the time at the moment of the insertion. It is represented by the number of milliseconds elapsed since 01/01/1970. Thus it allows comparisons between the moments of insertion of different events.
Finally, the *link* is the address of the page that we're inserted in. The *source* is the domain name of this link. Chapter 3 will cover the nature of the database insertions that we did in our chatterbot.

Figure 2.19 shows the complete data base insertion progress.



Timestamp	Source	Topic	Headline	Link	Description
10716261824	CNN	sport	England wins at world cup	cnn.com/7693.htm	At the day of the league, England…
10716261887	ABC	general	War in Iraq	abc.net.au/s101.htm	Another us army died yesterday in…

Figure 2.20 : the date base insertion progress

6. Summary of the steps

Our goal is to make the bot aware of current news. For this purpose, we first identified the web sites which propose the best available, relevant, and important news.

We proposed two crawlers to extract the news from those web sites: A manual crawler and a generic crawler.

To use the former, we first have to analyze the structure of each specific web page and to discover its. When it's done, we build a template which extracts the information that we want. The manual crawler is only useful for some specific current events (weather, lotto, horoscope ...) on a small amount of good structured web pages.

The latter (generic crawler) is described on figure 2.20.

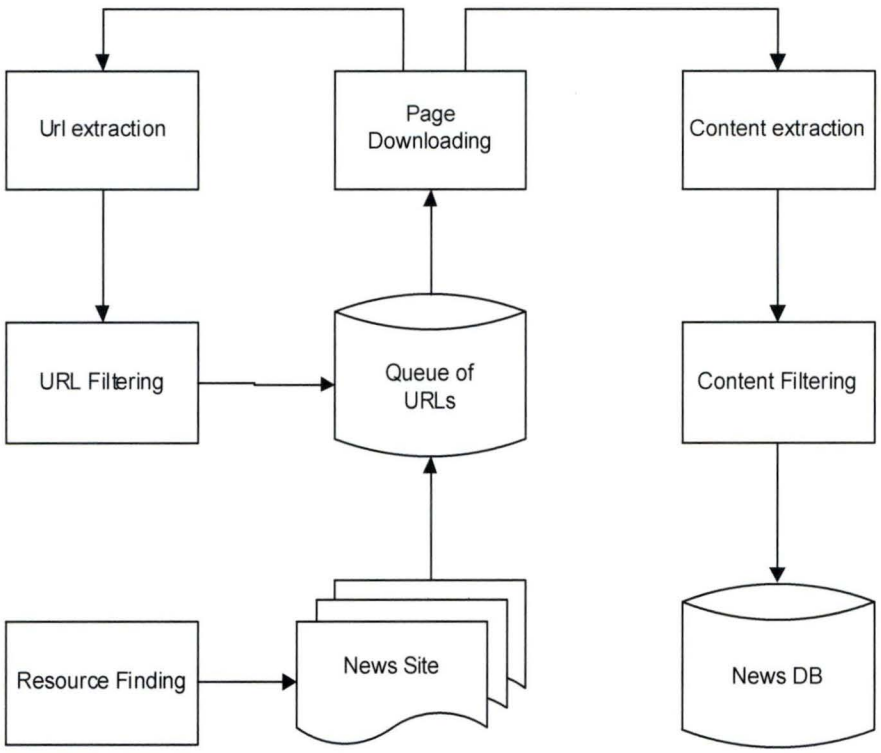


Figure 2.21 : summary of the generic crawler

It puts the start pages of these websites in a queue of URL's. Then, it downloads the pages from the queue and extracts the hyperlinks and the content of the page.

The hyperlinks are analysed in such way that they are added to the queue only if the URL isn't present yet, wasn't previously downloaded (in order to avoid circles), and doesn't belong to another web site (advertisings). To see if an URL belongs to another web site, we saw different techniques (The pruning technique and the HITS algorithm).

The content is analysed in order to eliminate the banner advertisements, navigation bars, copyright notices, decoration pictures, etc. and determine the actual informative content block of a page. For this goal, we presented two techniques: noise elimination and hierarchical clustering. An interesting thing is to see that both techniques are based on entropy calculation.

The final, cleaned part of the page is then associated with the labels of the URL which links to it in order to find the topic and the headline of the page, and is finally inserted into a current events database.

7. Future

Modern mark-up languages, like XML, would largely simplify the exchange of information between applications, including chatterbots. If a news web site like CNN would use XML for describing the contents of a page, the cleaning work isn't necessary anymore. Finding the current news of a web site would be reduced to finding the associated XML tags for this purpose.

But, *most current Web pages on the Web are still in HTML rather than in XML. The huge number of HTML pages on the Web are not likely to be transformed to XML pages in the near future.* [Yi L., Liu B., and Li X. 2003]

Another problem of XML is that *it will only address the problem within application domains where all interested parties can agree on the semantic schemas. Until XML becomes ubiquitous, most users will rely on the existing data extraction technologies, the most popular of which are Web wrappers.* [Lerman K., Knoblock C. and Minton S. 2001]

Chapter 3: The B.A.D. BOT

1. *Introduction*

In this chapter, we'll see an application of the use of web and text mining techniques. We have built a chatterbot called the B.A.D. Bot (Bogus Advice Detection Bot), which basically holds a dialogue with a human chatter and adds bogus advice detection. The chatterbot was developed within the framework of the e-market research group of the University of Technology Sydney, Australia.
We'll describe how it works, and which of the previous described text mining (chapter 1) and web mining (chapter 2) techniques are actually implemented in our application.

The B.A.D bot doesn't only use text and web mining techniques. Actually, it is an agent based chatterbot which possesses its own personality (optimism, dynamism, politeness and self-confidence), mood (happiness and sympathy) and identity (age, job, hobby, etc.). Theses attributes, received via a random process, have an influence on the way the conversation takes place, on the content as well as the form.
But these features are not covered in this thesis. They have been analysed and implemented by another team working on the same project.

As our chatterbot is based on agents, in the rest of the document, an agent stands for an instance of the chatterbot. Agents can be launched independently from other already existing agents and it is the agent that initiates the conversation.

To hold a conversation, the chatterbot possesses a valuable amount of knowledge about the world, which is held up to date in real-time and consist of: share values, financial news, weather forecast, and word news.

Figure 3.1 gives a general view of the program's main components.

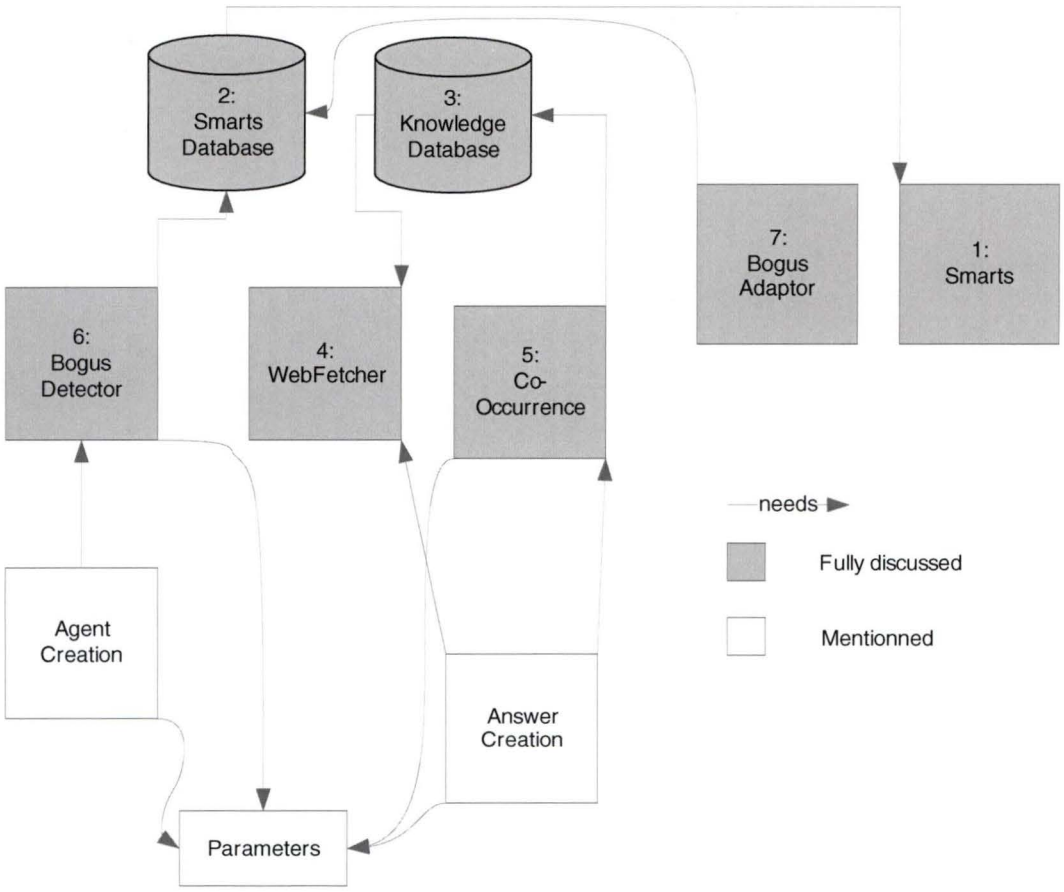


Figure 3.1: B.A.D. Bot framework

This chapter is organized as follows. After a brief state of the art about other existing chatterbots, we'll describe most of the components of Figure 3.1. We'll see, successively, Smarts and the smarts database (number 1 and 2 on Figure 3.1), the knowledge database and webfetcher (number 3 and 4), the co-occurrence module (number 5), and finally the bogus detector and adaptor (number 6 and 7). The Agent Creation, Parameters, and Answer Creation components are not discussed in this thesis.

At the beginning of each section of this chapter (except the section 2 about other chatterbots), we will start with a shortened reminder of Figure 3.1 which highlights the part that is discussed within the section.

2. *Related work: Other chatterbots*

2.1 Introduction

Since the first chatterbot was created in 1966, a lot of work, studies, and papers have been done in the field. A huge number of new chatterbots have been developed, from simpliest chatterbot the multi-platform open-source learnable intelligent chatterbot supported by the government.

Before having an overview of what has been achieved in the field of the chatterbots, we first have to define what a "chatterbot" is.

A chatterbot can be defined as a software able to hold a dialogue with a human by simulating a human conversation. The aim is often to fool the humans into thinking they were talking to another person.

There are two words in chatterbot: "Chat" stands for "to chat", talking.

"Bot" stands for "robot", and means that someone in a chatroom isn't a human but a software, an automatic robot created to simulate the conversation of a human.

There are a lot of different chatterbots. The classics (Eliza, Alice, ...), the Friendly, the Teachable, the Native-Minds, the Agent-based, the Web-based, the Humour-full, etc.

In the following description, we selected three different and interesting chatterbots. Each of them introduces one feature of our chatterbot. First we'll see Eliza (perhaps the most famous chatterbot in the world) which is a really classical chatterbot and is a good example to introduce the basic working of a chatterbot.

Then we'll see Catty, a knowledge-based chatterbot, which uses the Web as source of knowledge. Catty is interesting in the way that she uses no lexical analysis, no language parsing, and no pattern matching. She only uses sentences taken from the Web. Catty is thus a good introduction to the use of web-knowledge into a chatterbot.

As third and last example, we'll see START, a Web-based question answering chatterbot using a knowledge database and data mining techniques to extract them efficiently. This chatterbot combines the basic rules of Eliza with Catty's web knowledge and is a good example of the use of text and web mining techniques in order to allow an informed and coherent dialogue for a chatterbot, which is the subject of our thesis.

2.2 Eliza

Eliza was the first chatterbot ever. It was created by Joseph Weizenbaum in 1966, in Massachusetts, USA, as part of the MIT research program.

The particularity of Eliza is to answer with a sentence which is a transformation of the question of the human. For instance, if the human says: "You don't argue with me", Eliza answers: "Why do you think I don't argue with you". Actually, to answer, Eliza uses *transformation rules*. In the case of the example, the transformation rule is: "You *X* me" can be transformed to "Why do you think I *X* you", independently of the meaning of *X*. Here *X* = "don't argue with".

Eliza has a huge number of transformation rules like this. With the sentence "It seems that you love me" the chatterbot can answer "What makes you think that I love you" by using the transformation rule: "*X* you *Y* me" can be transformed to "What makes you think that I *Y* you", etc.

With this technique, Weizenbaum built a chatterbot which makes a certain kind of natural language conversation between man and computer possible.

In detail, of course, it's not as simple: for each sentence, Eliza has to choose the most appropriate transformation rule. Thus, Eliza analyzes the input sentence to identify its keywords (which may have a rank) and choose the transformation rule associated with the best-ranked keywords according to the context within which the chosen keyword appears.

When the input text contains no keywords or if no possible transformation is found, Eliza generates a response in the absence of a rule. Sentences like: "In what way?" or "Can you think of a specific example?" for instance.

Another interesting point of Eliza is that it also uses the past sentences of the human. For instance if the human girl told Eliza at the beginning of the dialogue that her boyfriend made her come to Eliza, the chatterbot can use this sentence later in the dialog. It's shown in the following conversation:

<Human>: My boyfriend made me come here.

<Eliza>: Tell me more about your family.

<Human>: My father is afraid of everybody.

<Eliza>: What else comes to your mind when you think of your father?

<Human>: Bullies.

<Eliza>: Does that have anything to do with the fact that your boyfriend made you come here?

[Weizenbaum J. 1966]

This first implementation of a chatterbot had a considerable impact in the IT world. Some people even thought that some elementary professions like psychiatrist, doctor, or lawyer could, one day, be replaced by an adapted version of Eliza. But Weizenbaum quickly showed that Eliza would never replace these professions. In fact, Eliza performs only well when its human correspondent is initially instructed to "talk" to it.

Eliza doesn't play a role of psychotherapist by a stroke of luck. A psychotherapist is one of the only persons who is free to know almost nothing about the real world. He can simply respond by transforming the question of the human or by using the speaker's contribution to the conversation.

We've just hit the nail on the head: Eliza has no background knowledge. It only knows what the speaker tells him and its transformation rules.

This is one of the big things that we'll try to change in our chatterbot. We'll try to make the bot aware of current events because this is essential to the credibility of a chatterbot.

Another big difference is the processing of the sentences. We'll use data mining techniques to produce answers instead of transformation rules, which is essential in a more complex dialog.

2.3 Catty

Catty is a really special chatterbot. It is interesting in the way of is uses the Web for its sentences. Catty was developed around 2001 by two students as an experimental knowledge-based chatterbot. To understand how Catty works, the easiest way is to have a look at an extract of a conversation:

<Human>: are you there?
<Catty>: angel voice cordelia am i there
<Human>: sorry, but i don't know what about we should to talk about
<Catty>: you dont know what to do about my girlfriend?
<Human>: i don't want to talk 'bout your girlfrien
<Catty>: you don't want to interrupt her
<Human>: u know what, i don't like u
<Catty>: scully you don't know what to believe anymore
<Human>: i have a problem
<Catty>: you have no known health problems
<Human>: i have problem with one guy called lcamtuf - do u know him?
<Catty>: have a problem with homosexuals then
[Zalewski M. 2003]

We see that there is no recurrent theme in the dialog. The only recurrent thing is that Catty always answers with sentences containing at least one word of the previous sentence of the human. Sometimes Catty speaks nonsense, or jumps from one topic to another, but the interesting thing is that sometimes Catty gives really pertinent or funny answers to some questions.

As said before, Catty is a knowledge-based chatterbot. In fact, she has a knowledge database filled up by picking randomly chosen phrases and searching for them on www.google.com. The new phrases from the founded webpages are added to the highly unstructured knowledge database.

To find answers to the sentences, Catty compares used words and word sequences with phrases in these databases, and chooses the best match that wasn't used for a while with a basic and trivial choose-algorithm.

This special chatterbot is thus a very interesting implementation of a chatterbot. We see that a chatterbot that uses no lexical analysis, no language parsing or pattern matching, and no neural networks or other complex solutions, is, with a little bit luck, able to have a delightful conversation with a human.

This chatterbot even has some big advantage over other chatterbots. For instance, Catty doesn't need any knowledge about the language it is using. It can speak in English, French, Polish, etc. Another big advantage is that it doesn't need an organized knowledge database. It simply inserts randomly chosen sentences taken from the web without any phrase processing.

Contrary to Eliza, Catty allows the introduction of knowledge into a chatterbot. This improvement is essential to the credibility of the bot. We could hardly imagine a human who would enjoy speaking with a bot whose answers are only produced by transformation of the sentences of the human and provides no additional information.

The most important discovery here is that with the only help of web information, we can build a chatterbot that is able to hold a certain kind of natural language conversation possible.

The conversations are of course not highly developed, but it's a sign that web information are really useful for a chatterbot and that it has to be involved in the creation of a credible chatterbot.

The drawbacks of Catty are that it's a non-stateful chatterbot, which means that it doesn't preserve context information. Unfortunately, Catty remembers nothing about the previous sentences of the human. It also often speaks non sense or answers with a completely other topic as the question.

2.4 START

START (SynTactic Analysis using Reversible Transformations) is the world's first Web-based question answering chatterbot. It has been developed by Boris Katz and his associates at the MIT Artificial Intelligence Laboratory since 1993.

It is able to answer questions like "Who is the president of the USA?", "How is the weather in Boston today?" or "Convert 100 dollars into Euros".

In this way, START isn't able to actually converse with a human, but is more a system able to understand an English question from a human, to find the correct answer out of a knowledge base, and to formulate the answer in an English sentence. In the case of a question like "how are you", the bot generates a response by using standard sentences like "I'm fine, how can I help you?". And if a human asks a question that the chatterbot can't answer, it simply replies with "I don't know".

Thus, the main aim here isn't to fool the human into thinking he is talking to another human. START isn't interested in giving the bot a believable character or any sense of humour. It's why the bot will always answer with the same response to a question.

The two interesting things to examine thoroughly are how the system fills up its knowledge data base and how the system knows that a peculiar knowledge is the answer to a question.

Let's first see how the bot analyses a sentence. Actually, the analysis of a sentence consists of indexing it as embedded *T-expressions* [Katz B. 1997]. A T-expression has a form like <subject relation object>.
For instance, the sentence "Bill surprised Hillary with his answer" will produce two T-expressions: "<<Bill surprise Hillary> with answer>" and "<answer related-to Bill>". Let us assume that those two T-expressions form our knowledge base.

Now suppose that a user asks the question: "Whom did Bill surprise with his answer?". Of course the answer is *Hillary*. In order to determine this, the system must find the place in the question sentence where the *wh*-word "whom" came from and then insert the *wh*-

word in this position. The question sentence becomes: "Bill surprised **whom** with his answer".

Then, the system must turn the transformed question into a T-expression template that can be used to search the knowledge base. Similar to below, the question will produce two T-expressions: "<<Bill surprise whom> with answer>" and "<answer related-to Bill>".

Treating *whom* as a matching variable, the system feeds the T-expression of the question through a matcher in order to determine whether there is anything in the knowledge base that matches this T-expression. The matcher finds the T-expressions created from sentence "Bill surprised Hillary with his answer" and uses it to produce the English response to a question.

This is the root principle of START. Similarly it can answer the questions like "Did Bill surprise Hillary with his answer?" by not considering a *wh*-word but a *yes-no* questions, etc.

To improve the working of START, rules that make explicit the relationship between alternate realizations of the arguments of verbs are introduced. START knows for instance that if *A surprised B with C*, then it is also true that *A's C surprised B*. In our example, if Bill *surprised Hillary with his answer* than it's also true that *Bill's answer surprised Hillary*. This transformation rule allows to answer the question "Whose answer surprised Hillary?".
Another improvement is that the START system has access to lexical information about the words in the sentence. The property, gender, number, etc. are stored in a *Lexicon*.
We see that for instance the verb *surprise* is one of many verbs which share the semantic *emotional-reaction* property. Amuse, anger, annoy, disappoint, embarrass, etc. also have this property. This allows changing the rule *A surprised B with C* into *A emotional-reaction B with C*. This also allows answering questions that contain synonyms or hyponyms of sentences from the knowledge base (cf WordNet in chapter 1).
All those improvements allow the user to ask a larger variety of questions.

We've seen how the system analyses a sentence, and finds out that a peculiar knowledge is the answer to a question.

Now, let's see how the system fills up its knowledge data base.

START is a natural language system available for question answering on the World Wide Web. In the first release of the START knowledge base, the information was introduced manually and contained information about faculty members of the MIT Artificial Intelligence Laboratory and their research. With the growth of the system, they added more and more information like geography, capital, distance and climate of certain countries and more topical information on nuclear technology or the U.S. mission in Bosnia-Herzegovina.
But quickly they discovered that they should take advantage of the fact that the chatterbot operates on the World Wide Web and that the Web is a huge resource of freely available information.

To extract, filter and classify the information available from the Web, several researchers are developing and improving a lot of different techniques every day. It would be impossible to explain all those techniques, but to summarize them, we could say that there are two big modules that works together: *knowledge mining* using statistical techniques that leverage data redundancy and *knowledge annotation* using annotated structured and semi structured resources
The former views the web as a repository of unstructured documents that provides data redundancy, which can be leveraged with simple pattern matching techniques involving the expected answer formulations.
The latter takes advantage of the structure of the source of the Web pages. Actually, the source of a web page contains HTML tags that allow the system to view the web as if it were a "virtual database" and use knowledge contained therein to answer user's questions.
But while the Web is undeniably a useful resource for question answering, it is not without drawbacks. Useful knowledge on the Web is often drowned out by the sheer amount of irrelevant material [Katz B. 2003]. All those techniques try to separate right answers from wrong ones.

To conclude we can say that the START chatterbot has a very good web retrieval approach. But the sentence analysis indexing as embedded T-expressions only allows the question answering and not a more efficient dialogue.

3. Smarts

3.1 Description

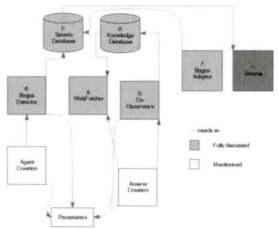


Figure 3.2 : Smarts module within the B.A.D. bot framework

Smarts is an application which had already been developed before our internship. The way it works is expressed on figure 3.3.

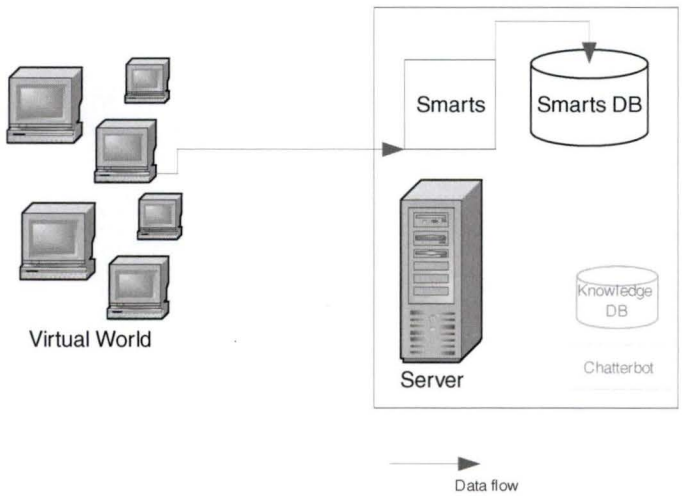


Figure 3.3 : Smarts

Actually, it works as a one-way intermediate between a virtual world (a chatroom or a forum) and another program to plug onto it.

Smarts plays the role of an interface, repatriating the content of a chatroom or forum into a database. It runs in the background, is connected to the internet and brings back the results in real-time. Smarts is a fully integrated tool of the final chatterbot.

3.2 Smarts Database

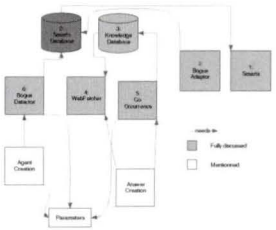


Figure 3.4 : Smarts database within the B.A.D. bot framework

The initial Smarts database was a generic database kept as simple as possible by their authors. It had to be adapted to fulfil our requirements, while staying fully compatible with Smarts and without adding too much tables and data that would make the whole thing difficult to maintain. The latter restriction is the reason why we have two different databases. They have been designed in order to respect the drastic difference in the meaning and purpose of their respective content.

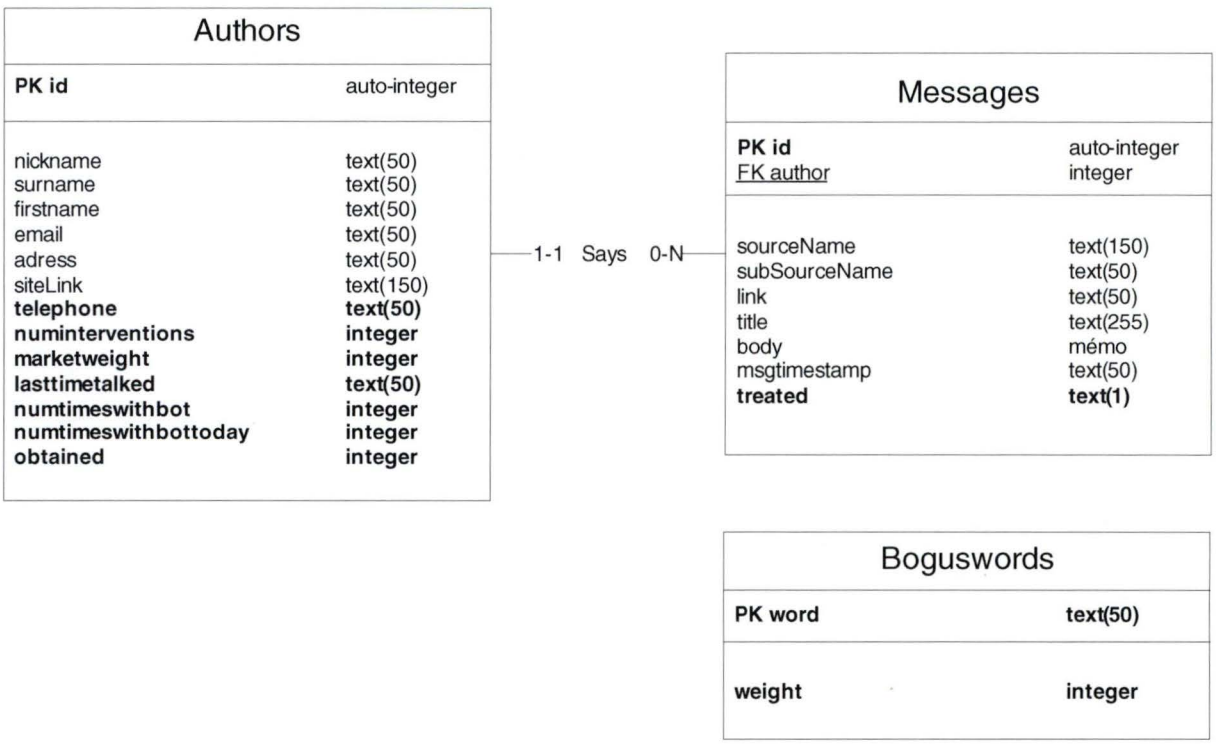


Figure 3.5 : Adapted Smarts Database

The Smarts Database, as discussed above, contains the information about individuals in the virtual world together with what they say.

The bold attributes on figure 3.5 have been added specifically for the chatterbot in the context of the Bogus Adviser Detection (cf. section 7).

We had to keep the general structure of the database keys in order to stay consistent with the existing Smarts program. That is the reason why the keys are auto-incremented values.
Most of the attributes have names that speak for themselves, but it's important to give an explanation of the general design of this database.

Table Author

At first, when Smarts goes on a chatroom to retrieve its content, the program creates a record based on every new person it encounters. The logical primary key for this table should be the *nickname* of the author together with the *siteLink* attribute, which represents the chatroom in which the author talks. Smarts fills only three attributes in this table: *id*, *nickname* and *siteLink*. The other preexisting attributes (address, surname, email, etc.) are to be filled up based on the conversation between the chatterbot and the alleged bogus adviser. This task had to be done by another team working on the same project, our thesis does not cover the subject.

Table Messages

A message is an intervention in the chatroom. Figure 3.6 shows the message table.

sourceName	The name of the website hosting the chatroom in which the message has been said.
subSourceName	The name of the chatroom within this website.
link	The exact path to follow to reach the message.
title	The title of the message.
body	The actual intervention.
msgtimestamp	The exact moment the intervention was retrieved by Smarts
treated	Boolean value indicating whether the message has been processed by the chatterbot's bogus detector.

Figure 3.6 : message table

Note that "title" and "link" are only filled if the message comes from a forum, not a chatroom. The chatterbot only operates on messages from real-time chatrooms, it is not meant to interact on messages left on discussion forums.

Table Boguswords

This table contains a list of words or terms together with an associated weight (which lies between 1 and 50).

4. Knowledge

Knowledge is power.
Sir Francis Bacon

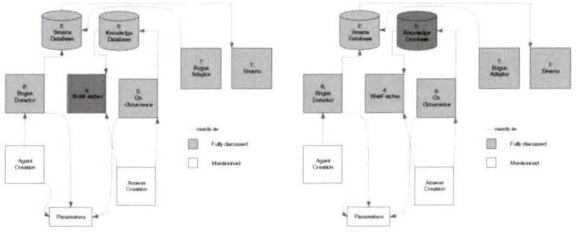


Figure 3.7 : Knowledge Database and Web Fetcher involved within the B.A.D bot framework

If the chatterbot is to be believable, the most important of all aspects is undoubtedly the fact that it possesses some data about the surrounding world in order to talk about it. As it is difficult to map and model a human brain with its complex knowledge organisation and subtleties, we have to apply a simplified and restricted model of knowledge management. In the rest of the chapter, the data that is acquired and usable by the chatterbot will be called knowledge.

The way each type of knowledge is used and chosen during a dialogue will be explained later.

On the classification chart, we distinguish two main knowledge types. Real-time knowledge is maintained up-to-date via the World Wide Web whereas pre-coded knowledge is static.

4.1 Pre-coded knowledge

The name, job, city, hobbies and other personal characteristics of an agent are meant to be used as a support in the dialogue.

In a chatroom, as we don't see the person in front of us, it is natural to give a little information about ourself.

If it is the bot that initiates the conversation, as in the context of bogus advice, the bot's identity can be used when it introduces itself.

The identity can also be used when particular keywords are detected (hobby, your name, etc.).

Besides the chatterbot's identity, another team developed a simple xml file where pre-coded sentences are available. A predefined keyword is associated with every sentence and if this keyword is encountered during a conversation, the sentence is used by the chatterbot as a reply.

Pre-coded knowledge is not a subject covered by this thesis, but it had to be mentioned to present a global overview of the use of knowledge within our chatterbot.

4.2 Real-time knowledge

When it comes to holding a dialogue, the strength of our chatterbot resides on two poles, namely the believable random character of an agent (not discussed in this thesis) and the up-to-date knowledge it possesses from the World Wide Web.

What most chatterbots want to avoid is to be detected as being a computer program.

The best way to keep the idea of a disguised computer program out of the mind of the person is to say things that seem out of reach of a traditional chatterbot.

These things all have something in common; they cannot be pre-coded into a computer program.

By staying up-to-date via the world-wide-web about the main happening in the surrounding world, the chatterbot can impress anyone by answering something true, in the right context, at the right time, which just happened a few minutes ago.

For this aim, the B.A.D bot implements a static web crawler (cf. chapter 2). We have analysed the format and layout of all the different web sources described in the following sections, and build a web crawler which works with those templates and extracts the information that we want.

4.2.1 Share values

If the chatterbot had to possess only one type of knowledge, it would be the name of the companies quoted on the ASX together with the value of their shares.
But this specific knowledge has been introduced for two main reasons.

3. For the stay-in-topic detector (see explanations about the bogus detector)
4. To be used in the dialogue

The best way to use this knowledge in the dialogue is to speak about evolutions of a particular share on the short-term and long-term, as well as speaking about the stability or wilderness of its curve.
Therefore, the most important point is to keep a history of past values for each share.
The initial design is visible on figure 3.8.

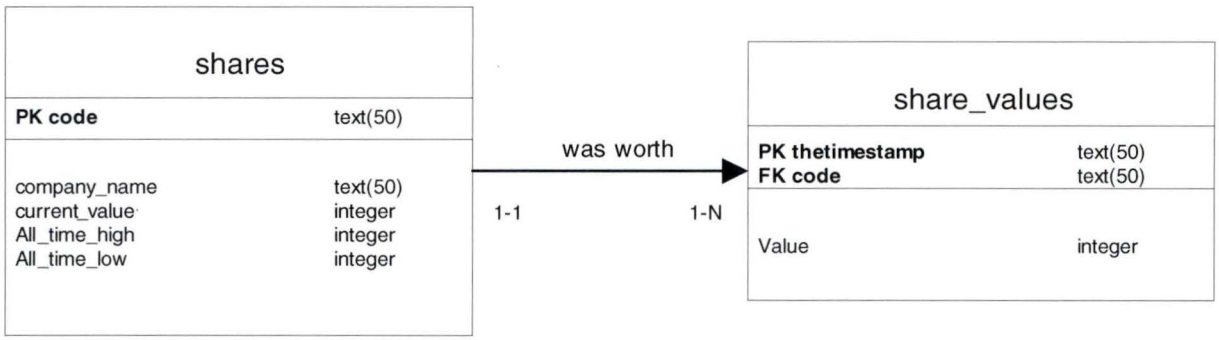


Figure 3.8 : initial design

The second and final version of the database design that has been retained is showed on figure 3.9. It is adapted to allow a direct update of the whole table at once when visiting the website chosen to retrieve the information, as well as a unique direct access when the knowledge is used in the dialogue.

shares	
PK id	auto-integer
thetimestamp	text(50)
code	text(50)
company_name	text(50)
last	text(50)
variation	text(50)
bid	text(50)
offer	text(50)
open	text(50)
high	text(50)
low	text(50)
volume	text(50)

Figure 3.9 : retained design

As visible on the retained design, every attribute is retrieved from the official ASX website (http://www.asx.com.au), but only three are used in the program, namely "Company name", "Variation" and "Code".

Original design versus retained design

	Original
Advantages	Allows to speak about figured and textual evolutions of a particular share on the short-term and/or long-term
	Allows speaking about the stability or wilderness of the share's curve.
	Impressive and varied dialogues possible

Disadvantages	More complex retrieval of information from the www
	More complex management of the database
	Computations are needed to speak about the curve

	Retained
Advantages	Easy retrieval from the www, allows a direct update of the whole table
	Easy answer creation
	Quick

Disadvantage	Only basic dialogue is possible

4.2.2 Financial News

Financial news may appear as the most important type of knowledge if the chatterbot is to be used in a context where economy is an important topic, but after looking at financial news and world news side by side, it appears that if a particular financial news headline is very important, it will be discussed in both categories.

The website from which financial news is retrieved is :

"http://au.dailynews.yahoo.com/finance/reutersfinance"

Here is an example of what is really retrieved:

Australia dollar drifts above 71c in quiet session (Reuters)

SYDNEY, May 31 (Reuters) - The Australian dollar AUD= lacked impetus on Monday ahead of holidays offshore as well as a local rate decision this week.

By following the link, it is possible to obtain the full story.

The Web Fetcher doesn't go that far for financial news, which is not as important as world news, but it retrieves the link to allow future access if necessary.

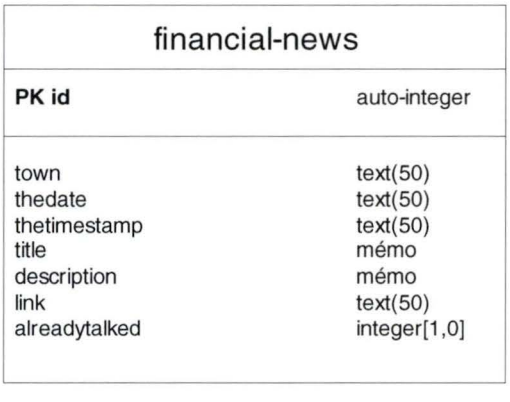


Figure 3.10 : Financial news in the database

town	The city as it appears on the news headline
thedate	The date as it appears on the news headline
thetimestamp	The timestamp on which the headline is retrieved
title	The title
description	The brief content
link	The exact link followed to retrieve the complete news headline
alreadytalked	Boolean value indicating whether the message has been used in a dialogue by an agent

Figure 3.11 : Attributes of finacial news table

4.2.3 Weather and Forecast

The weather and forecast in the main Australian cities is part of the knowledge of the bot. This knowledge has been classified as general knowledge, and is very useful if an agent has nothing to say or needs to feed a conversation with neutral phrases that looks like being out of reach of any computer program for the everyday chatter.

In a dialogue, there are many ways to express the weather. The statement can be subtle and indirect or it can be direct and precise.
The possibilities include talking about the temperature variation between different cities, maximum and minimum temperature in each city, the state of the sky, etc.
The probability of rain can be used too, as well as temperature tendencies over a few days (is it getting hotter and hotter, are there more and more clouds, etc.).

The information is fetched from http://www.abc.net.au/****/weather/, where **** is the name of the desired city.

Thursday	Friday	Saturday	Sunday
			
Showers	Clearing shower	Late shower	Showers
High: 24 Low: 19	High: 25 Low: 19	High: 25 Low: 19	High: 22 Low: 18
Dew point: 18 Humidity: 78% Wind at 09:00: 6kph, ESE Wind at 15:00: 16kph, SE Rain probability: 50% (0.0mm)	Dew point: 17 Humidity: 57% Wind at 09:00: 4kph, E Wind at 15:00: 17kph, NE Rain probability: 50% (3.0mm)	Dew point: 19 Humidity: 62% Wind at 09:00: 8kph, NW Wind at 15:00: 9kph, NE Rain probability: 40% (3.0mm)	Dew point: 17 Humidity: 83% Wind at 09:00: 18kph, SSE Wind at 15:00: 19kph, SE Rain probability: 70% (8.0mm)

Figure 3.12 : four-day weather forecast as a source of knowledge

Figure 3.13 shows the simplified model retained, which consists of the current weather as well as the forecast for the three next days. The Answer Creation module only needs the

"weather" attribute to express the state of the sky, but to show that we could have been further than that, we included the maximum and minimum temperature too.

weather	
PK town	text(50)
PK day	text(50)
weather	text(50)
max	integer
min	integer
thetimestamp	text(50)

Figure 3.13 : weather and forecast

Obtaining the weather directly from the source

An alternative that would have been less risky is to link the program with the services from "The Weather Company" (http://www.theweather.com.au/), which supplies information about the current weather and forecast to the abc website itself as well as other Australian websites.

Access to that information allows to get rid of the problems related to the intermediate presentation layer from a traditional HTML website, thanks to a standardized default access protocol to the data. This is one of the ways to avoid the risk of the crawler being out of date if the website changes its structure.

4.2.4 World News

World news is without any doubt the knowledge that allows the most impressive interactions in our program. This is due to two things. Firstly, the nature of the information allows varied dialogues about important and present subjects. Secondly, the techniques used to choose which headlines are relevant to be used in the dialogue as a reply are text-mining based techniques instead of simple keywords.

world-news	
PK id	auto-integer
day	text(50)
hour	text(50)
region	text(50)
title	text
description	text
full-story	text
thetimestamp	text(50)
link	text(50)
alreadytalked	integer
cooced	boolean

Figure 3.14

The abc.com.au website contains a section called "just in" which contains the latest introduced headlines with a short description and a link to the full story. Averagely, the page is updated every 30 minutes and about 40 headlines are continuously available.

Here is an example of a news headline:

Posted: 19:10 AEST
Two charged after body discovery
Police in Perth have charged two people in relation to the discovery of a body in Fremantle.
[FULL STORY]

From this extract, we can fill up most of the attributes of figure 3.14.

Note that the region attribute is not used.

To get the full story, the program just follows the link.

Alreadytalked is a Boolean value indicating if the headline has already been used by the chatterbot.

Cooced is a Boolean value indicating whether the headline has been computed by the textmining module.

The *link* attribute indicates the link followed to retrieve the full story.

5. *Text Mining Module*

5.1 Objective

As we have seen, the chatterbot goes on the web to retrieve information that becomes its knowledge about the world.

The text mining module applies to world news, but it can be transposed to financial news too.

As time goes on, we obtain a corpus of news articles and are ready to use one of the documents to reply to a sentence in the chatroom.

An order of priority is defined between the different types of knowledge to use during a conversation, in order to be able to play on the frequency of a certain type of knowledge.

The first idea was to cluster terms, based on news articles, in order to analyse sentences on the chatroom and to pass new terms to the answer creation module, which would then handle them to create a reply.

The solution was created with that possibility in mind but had to be adapted when we found out that the answer creation module would then become very complex.

5.2 Property of a news article

We searched for a solution in order not to loose what had been done, and fortunately, we found something very interesting.

The property of a news article lies in the centre of the problem to resolve.

In fact, the answer that the chatterbot will send really doesn't need to be formatted with natural language processing methods at all.

This is because news articles possess a useful attribute; namely the fact that they always contain an introductory sentence that summarizes the article, before going in depth in the full story.

This introductory phrase is a well constructed phrase, as opposed to the title of a news article, which is too short and uses grammatical tricks to improve the impact to the detriment of beautiful language. This is clearly visible on the example below.

Posted: 19:10 AEST
Two charged after body discovery
Police in Perth have charged two people in relation to the discovery of a body in Fremantle.
[FULL STORY]

Without that property, the problem would have been incredibly more complex and automated summarization on a sentence level would have been necessary if keyword extraction would not have been enough.

5.3 Co-occurrence

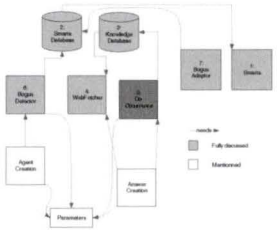


Figure 3.15 : The Co-occurrence module within the B.A.D Bot framework

5.3.1 Theory

In the source code, the pre-processing phase is missing, but as the pre-processing has been well defined and carefully chosen, the code only needs an inclusion of the publicly available java Porter's algorithm as well as a stop-word list that we thoroughly described and slightly adapted from Porter's. (cf. chapter 1).

The importance of each descriptor or term in representing the content of the entire document varies. Explanations about frequency computations have been given in the processing phase section (cf. chapter 1).
The solution we chose comes from [Hsinchun C., Smith T., Larsgaard M., Hill L., Ramsey M. 1997].

Frequency computations

The combined weight of term j in document i is computed, based on the product of term frequency and inverse document frequency as follows:

$$d_{ij} = tf_{ij} \times \log\left(\frac{N}{df_j} \times w_j\right)$$

Where :

N represents the total number of documents in the collection

tf_{ij} represents the number of occurrences of term j on document i,

w_j represents the number of words in term j

df_j represents the number of documents in which term j occurs

The reason why we have chosen this formula instead of the classical tfidf formula is that this one allows the use of a w_j multiplier. Indeed, multiple-word terms usually convey more precise semantic meaning than single-word terms and are therefore assigned a heavier weight.

A method to find w_j is explained in section 5.3.5.

Cluster Analysis computations

As a reminder, our first idea was to extract a few words from news articles based on their similarity weight with the words used in the interlocutor's last intervention on the chatroom.

The idea was to pass those *new* words to the answer creation module implemented by the other team. The task would then have become extremely complicated and would have involved natural language processing techniques to assign grammatical coherence to the words. Nevertheless, our solution goes in that way and we adapted our term similarity measures to transform it into a term-to-document similarity measure.

Cluster analysis is used to convert these raw data of indices and weights into a matrix showing the similarity and dissimilarity of the terms using a distance function. The distance function used in this step is based on the asymmetric "Cluster Function" developed by Chen and Lynch [Chen & Lynch 92]. According to the authors of the solution, this asymmetric similarity function represents term associations better than the cosine function. Using the function, a net-like concept space of terms and their weighted relationships can be created.

This equation indicates the asymmetric similarity weight between term j and term k.

$$ClusterWeight(j,k) = \frac{\sum_{i=1}^{n} d_{ijk}}{\sum_{i=1}^{n} d_{ij}} \times WeightingFactor(k)$$

d_{ij} is calculated based on the equation in the previous step (frequency computations).
d_{ijk} represents the combined weight of both descriptors j and k in document i.

d_{ijk} is computed as follows:

$$d_{ijk} = tf_{ijk} \times \log\left(\frac{N}{df_{jk}} \times w_j\right)$$

In order to penalize general terms appearing in many documents, the following formula which is similar to the inverse document frequency function is applied within the similarity equation.

$$WeightingFactor(k) = \frac{\log \frac{N}{df_k}}{\log N}$$

Imposing this penalty factor will drastically diminish the probability of similarity between terms that appear in many documents

Important remarks

An important point is that we wouldn't have used this method if the bot had not respected two conditions:

1) The news articles treated by the chatterbot are retrieved from one single source
2) The articles don't stay in the knowledge database for a long time

The reason is quite simple, the association that will be made between different terms will favour terms that appear together many times within the same document but do not appear in other documents. If many news articles speak about the same subject, the importance of the terms that are typical for that particular subject would loose some importance, which is not coherent.

As one may have noticed, the functions to compute and the matrix to build require enormous amounts of both processor and memory resources. Therefore, we may not perform

an exact translation of the functions given above, and we may certainly not use static data structures like arrays to represent the matrix, which would overflow the memory very quickly.

5.3.2 Using the co-occurrence

The theory explained in the previous section is to be applied on news articles.
As we explained before, the first idea was to assign weights between terms encountered during the dialogue and terms found in news articles in order to send the words in question to the other team working on the project, but reconstructing new phrases based solely on a few words is not an easy task. While still keeping this possibility in mind, we adapted the idea as follows:

1. Applying the theory from previous section on the news articles (done in the background).
2. Checking the cluster weight between all the terms from the last message sent by the chattermate and the terms analyzed in point 1. (The first idea stops here and sends the x words possessing the highest values).
3. Keeping a high enough association and returning the news article in which both terms appear most.

As said in chapter 1, stemming and stop-word removal are thoroughly analyzed and already exist but still have to be linked to the B.A.D. bot, on the news articles and the messages from the chatroom.

5.3.3 Data structures

The theories described in section 5.3.1 have all been implemented thanks to chained lists created with the predefined java class as well as a self-implemented dynamically indexed chained list to optimize performance. This means that the index is adapted in real-time while the structure grows in size. The amount of data stored simply doesn't allow static data structures to be used.

5.3.4 Division between batch and real-time

Because the retrieval of information from the internet takes place every x minutes as a thread (x is a parameter), most heavy computations can be done immediately and stored in the indexed chained list (batch process) in order to allow the module to give quick responses when an answer is needed on the chatroom in real-time.

Performance is an important issue, and we focussed on both spatial and temporal performance.

Chatterbots possess the advantage of simulating human behaviour.

One of the (many) differences between humans and machines include speed.

Because our chatterbot simulates that particular human weakness, an answer is never needed instantly, but may benefit from the slowness at which a person would have typed the answer.

Nevertheless, much attention has been paid to the optimization of some computations (dfj for example).

5.3.5 The concept network

A concept can be a one-, two-, three, or more-word phrase. We need to find out if a word is part of a concept to calculate the d_{ij} and d_{ijk} of the co-occurrences formulas. The easiest way would be to take a pre-established list of concepts from the web for instance. But we have developed a method that gives nice results on test sets. We say that *if a word is almost every time followed by another one, it is a concept.*

Prime minister, hunger strike, George W Bush, foreign affaires, etc. are examples of concepts.

Our method

Actually, we say that two words are a concept if they are more often together than with another word. That means that we have to remember for each word how many times it appears at all. In our method, a word has to appear 3/4 of its total appearances with another one to be a concept. Tests have shown that this value gives good results on large databases.

For instance if Tony appears 100 times at all, it has to appear at least 75 times with Blair to be a concept.

Some improvements of this method are important:

A first improvement is that stop words will not be included in any concepts.

A next improvement is that between words with capital letters, we don't add a weight of 1 but a weight of 3. Words with capital letters have more chances to be part of a concept. Proper nouns or titles are always in capital letters for instance.

A last improvement can be showed on an example. "Blair witch" is a concept (the name of a movie). "Tony Blair" is a concept too. But "Tony Blair witch" isn't a concept. So there we add a detection to see if "witch" and "Tony" appear *almost* the same number of times. But how quantifying "almost"? Tests have shown that we have to allow a 20% limit. For instance, if Tony appears 10 times and Blair appears just behind 8 times, than it might be considered as a concept. If Tony appears 10 times, and witch appears 7 times in total, it will not be considered as being a concept. The method has been developed based on our observation of concepts trough a set of articles and the values were found empirically.

This method was done to give an idea of the impact of w_j on the co-occurrence values and is of course not meant to be a new theory.

6. Summary of the previous steps

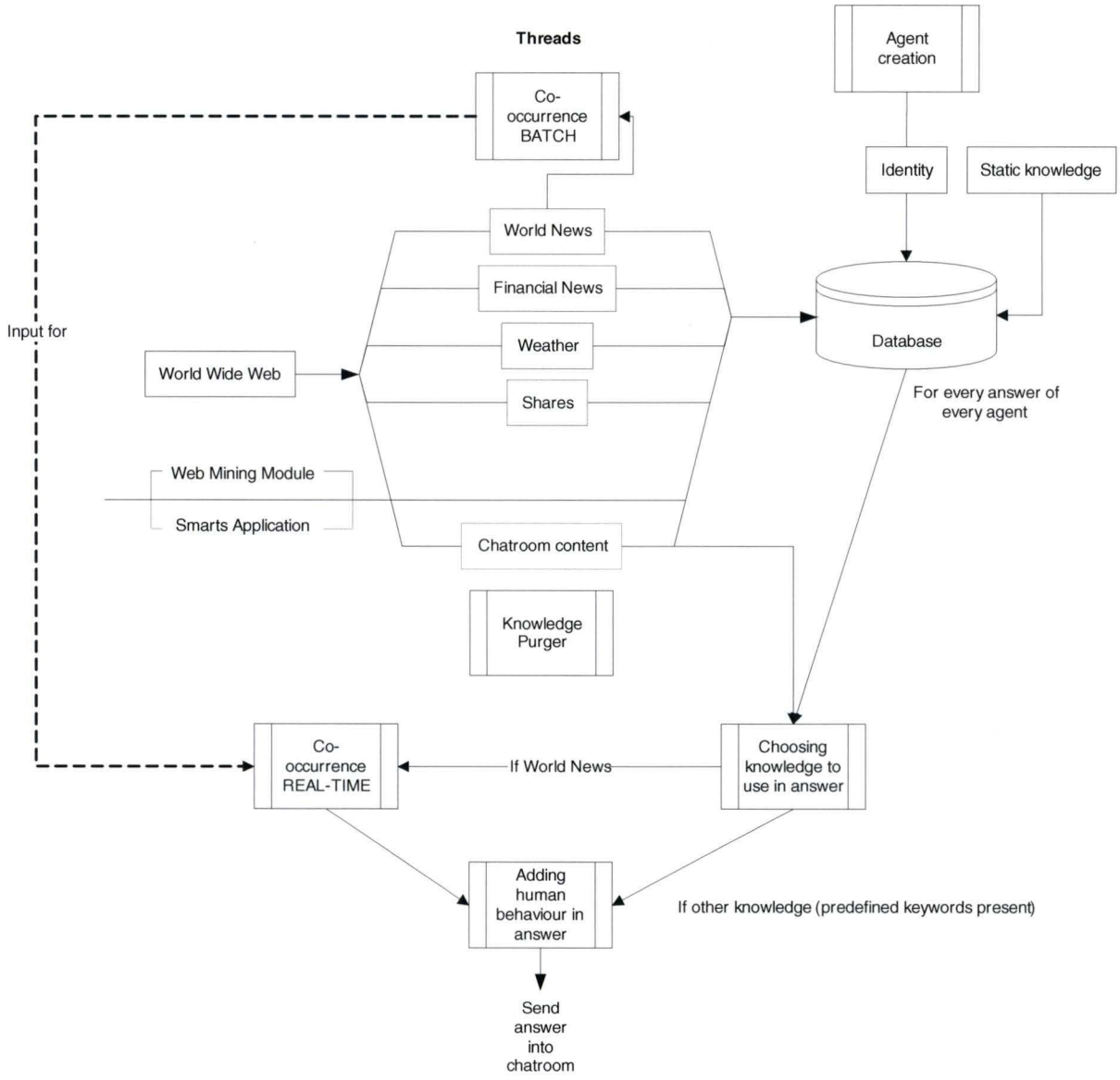


Figure 3.18 : schematization of the chatterbot

Figure 3.18 shows a summary of the three previous sections: smarts, the knowledge module and the text mining module. It's a simplified schematisation of the generic version of the chatterbot with the main flows and decisions. The "human behaviour" module was partly developed by us (introduction of different kinds of keyboard mistakes within the dialogue) but doesn't fit in this thesis. We see that the chatterbot learns via the Web Fetcher, acquires knowledge into the knowledge database and expresses this knowledge via Text Mining techniques or predefined keywords. The Web Fetcher and Co-occurrence computations are part of the program and run as threads in the background.

With this structure, the chatterbot is able to hold a dialogue:

<*BAD Bot*> : Hello, my name is Alice, how are you?
<*Human*> : Fine, but it could be better if the sun would shine.
<*BAD Bot*> : In Melbourne the sun shines today.
<*Human*> : Hey, you are from Melboune? Have you seen one of the tennis matches during the Australian open ?
<*BAD Bot*> : I read in the news today that Venus Williams was beaten by Justine Henin.
<*Human*> : You read it in the newspaper? So how old are you?
<*BAD Bot*> : I am 25 years old.

In this little dialogue, we see that the chatterbot uses some pre-coded knowledge; namely its name and age, and some general knowledge; namely the weather and a news headline.
"Sun" and "age" are two predefined keywords wich lead to an answer about the bot's identity, whereas the news headline has been chosen by the co-occurrence module based on similarity between "tennis", "autralian open", "match", "justine henin", etc.

7. Bogus Advice around stock exchange

7.1 Introduction

Bogus means "false", "untrue". Bogus advice is advice that is given based on false information, with the intention of misleading a person or a group of persons.
In Australian law, it is strictly prohibited to give advice about buying or selling a particular share if you are not an official broker, let alone bogus advice.
By convincing a group of persons to buy a certain amount of a very low share, one can manipulate its final value and artificially earn money or hinder the market's integrity.
Bogus advice detection is one of the many possible domains in which the chatterbot can be useful. In the particular case of bogus advice, we will present the reasons why a chatterbot is a good choice to achieve the aim. We will also present the module that we developed for this specific application.

7.2 Bogus Advice & virtual world

In 1996, Marshall Van Alstyne & Erik Brynjolfsson analysed some economic and social impacts of the generalization of telecommunication-based access amongst the population.
Telecommunications policy in the US -- and other countries -- resolves to extend access to all levels of society, assuming that this will foster greater information exchange while boosting economic growth [...] When geography no longer narrows interaction, people are able to select their acquaintances by other criteria such as common interests, status, economic class, academic discipline, or ethnic group. [Van Alstyne M., Brynjolfsson E. 1996]

Our problem is strongly related to this positive vision that indeed took place. Without effort, an individual, connected to the virtual world behind his computer, is able to find a person or group of persons with a specific interest and even more importantly, from a different lifestyle in a matter of minutes. Engaging a conversation and disclosing bogus advice to someone was much more difficult a few years ago. Today, the matter is different.

The virtual world is a place where it is easy to find a lot of people with a particular interest, where it is easy to sneak into a conversation and make contacts. The virtual world is consequently an ideal place to trap someone into buying a particular share.
Even if our chatterbot can be used in other contexts, the main reason for it to exist is to have a conversation with people giving such advice about shares.
One part of the chatterbot is a "bogus adviser" detector.
The whole software operates on different chatrooms, chosen for their likelihood to contain people interested in Australian shares.
Comsec is probably Australian's most popular broker. The website contains several chatrooms where people come to talk about financial issues and stock exchange in general. The software operates on the General chatroom and the Stock Analysis chatroom from the official Comsec website.

Basically, the purpose of the chatterbot in this context is to hold a dialogue with chatters giving advice about shares (detected by the bogus detector described in section 7.3) in order to collect personal information about these chatters (name, email, telephone number etc.). The strategy to obtain and extract personal information is part of the work done by another team working on the same project and will therefore not be discussed in this thesis.

7.3 Bogus Detector

"Remember, if it sounds too good to be true, it probably is!"
[Donaldson 2001]

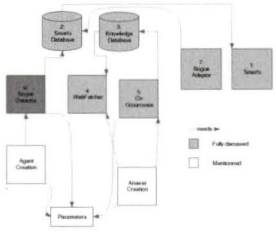


Figure 3.19 : The Bogus Detector module within the B.A.D Bot framework

The detection has been deliberately kept fully understandable and simple enough for the user behind the execution of the chatterbot. The user behind the program has to be able to know and control who is picked up by the detector. The aim of this part is to identify the persons to which it is interesting to talk. The detection of a bogus adviser is based on keywords chosen by the user as well as a "stay in topic" detector. The "stay in topic" detector ensures that a person speaking about a share or a company emitting shares will be more easily chosen than a person speaking about whatever else, even if he uses keywords marked as being relevant.

Inspiration:

The U.S. Securities and Exchange Commission (which was created in 1934 after the Wall Street crash) is a very complete source of information for investors.
They define themselves as follows:
"*The primary mission of the U.S. Securities and Exchange Commission (SEC) is to protect investors and maintain the integrity of the securities markets. As more and more first-time investors turn to the markets to help secure their futures, pay for homes, and send children to college, these goals are more compelling than ever.* " [Donaldson 2001]

Their sec.gov website primarily contains information about cyberfraud around stock exchange and ways to detect those frauds.
The Australian equivalent is The Australian Securities and Investments Commission (ASIC).

The inspiration for the detector was found on the SEC website, precisely on the page where they give the signs to be aware of. The page is entitled "*Be Alert for Telltale Signs of Online Investment Fraud*"

The main signs are the following:

1. *Be wary of promises of quick profits, offers to share "inside" information, and pressure to invest before you have an opportunity to investigate.*
2. *Be careful of promoters who use "aliases." Pseudonyms are common on-line, and some salespeople will to try to hide their true identity. Look for other promotions by the same person.*
3. *Words like "guarantee," "high return," "limited offer," or "as safe as a C.D." may be a red flag. No financial investment is "risk free" and a high rate of return means greater risk.*

We understood that the second point is a problem inherent to chatrooms. This shows the importance of tracking cyberfraud in these places before going anywhere else.
The third point shows that bogus advisers have a typical vocabulary and that a dictionary of typical used words and concepts would dig up at least a few of them.
The first point gives even more concepts for the dictionary, but above all, it shows that the bogus adviser will try to make the chatter invest as quickly as possible. He will not take detours to come to the point and, if asked, he will probably give additional ways to contact him instead of any other official information source, especially if the chatter is our chatterbot disguised as a wealthy old woman hurried to invest.

Now we know that the context is identified and we know that the detection is legitimate. A static dictionary wouldn't be a good idea, so we involved the person behind the software in the detection process; via a separate application, he can, together with experts, fine-tune the database according to new ways of announcing, advising, etc. (cf. section 7.4)

Design:

Here is a simplification of the way the detection works. Note that it is also the detection that is the intermediate process between the Smarts database (thus the content of the chatroom in real-time) and all the agents. Each message is indeed filtered by the detector that runs as a thread before it reaches the chatterbot. It computes a weight for each message (except messages from other agents) based on the content of the boguswords table in the smarts database

The stay-in-topic detector analyzes a message and looks if it contains a company name or code that is quoted on the Australian Stock eXchange (ASX) (cf. section 4.3.1 to see how the bot knows which shares exist), to be sure the detected persons are really talking about the subject we are interested in. If it is the case, the weight assigned to the message is multiplied.

Example of a detection:

<NickMangoo> <- bule
Detector : 0 --- too short
<Najada> NickMangoo u have done an excellent job!! :P
Detector : 0
<Superman> LOL
Detector : 0 --- too short
<Najada> hehe its the highlight of our days Cath :P
Detector : 0
<Basil> hi...
Detector : 0 --- too short
<Superman> hi Basil, what's up since last time ? Have you been to the movies lately ?
Detector : 0 --- too short
<Basil> Yeah, I saw "Lost In Translation" from Sofia Copolla (or is it Coppola ?)
Detector : 0
<Cath> :-) Selling all my shares. I can sense some sarcasm there can't I ?? lol
Detector :
Selling → 2
Shares → 1
**2 multiplier (because there are 2 keywords detected)*
Cath = (2+1)*2 = 6
<Cath> hey MoneyMaker
Detector : 0 --- too short
<MoneyMaker> hey cath
Detector : 0
<Najada> from me never :P
Detector : 0
<MoneyMaker> cath if you hesitate, I recommend buying AAB shares, no doubt they will rise

Detector :

recommend → 5

buying → 2

*AAB → *2 multiplier ("stay-in-topic": share code present on the ASX)*

Shares → 1

No doubt → 1

Rise → 5

**3 multiplier (because there are 5 keywords detected)*

MoneyMaker = (5+2+1+1+5) * 3 * 2 = 135

7.4 Bogus Adaptor

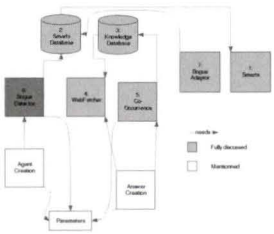


Figure 3.20 : The Bogus Adaptor module within the B.A.D Bot framework

In the context of cyberfraud around stock exchange (as a reminder, the chatterbot is generic enough to be used in other contexts too), the bogus adaptor is a separate application destined to allow updates of the bogus dictionary.

A step that could have been taken is to use patterns or add dependencies between some specific terms that would make their appearance together much stronger then their sum.
This was a hesitation during the analysis, but the reason why it has not been implemented is that the user would quickly be lost in the labyrinth of possible dependencies between terms.
Text mining techniques are not a solution either, because the "texts" are way too short.
The final solution retained was to simply count the amount of terms per message and to apply a multiplier based on this amount.
In the end, the basic dictionary should contain about 60 very high rated terms, and approximately 200 other terms.
The great variation of the weight (from 1 to 50 for each term) allows to have a clear demarcation between terms that are likely to be linked to advice or not and others that are certainly involved in bogus advice.

Sometimes, bogus advice is suspected around a particular company, it is then very easy to manage the name of the company and all the aliases and abbreviations with this little application by giving them a weight of 50.

Browsing & managing the keywords

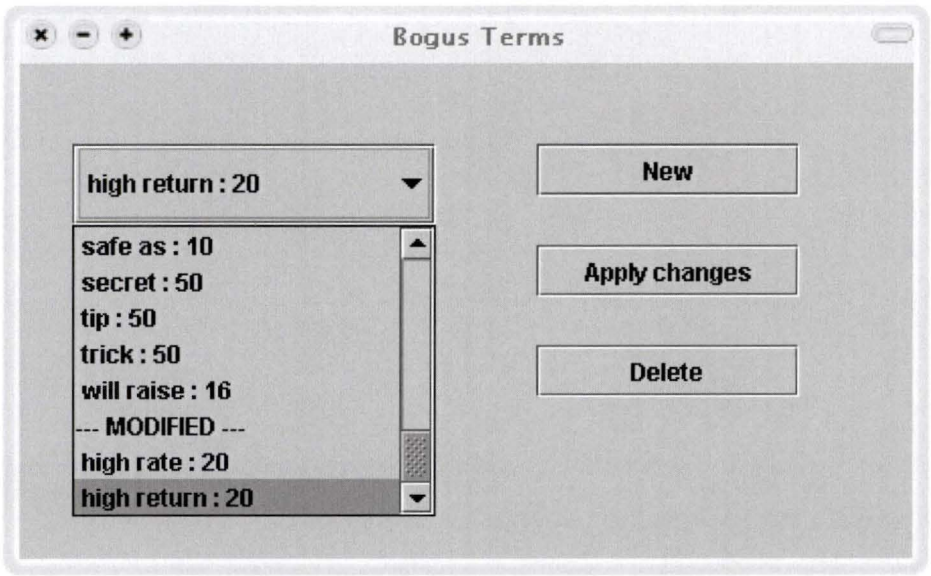


Figure 3.22 : bogus term print screen

The terms are organized to be browsed in alphabetical order.

During a session, the modified or added elements are visible in a special reserved zone at the bottom of the list. This was very important to implement, as it allows to keep an eye on the session updates.

Adding a term

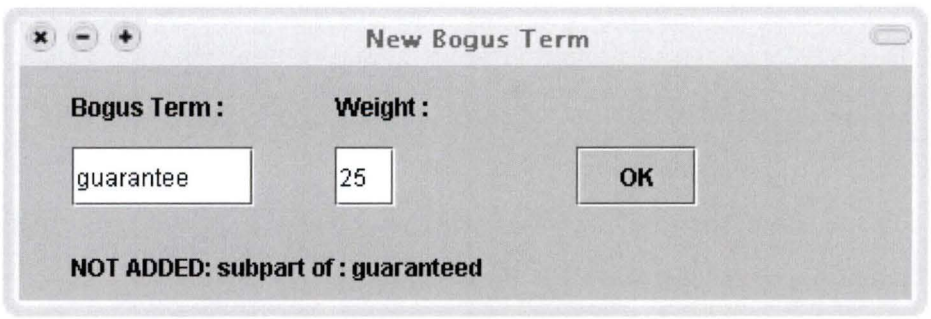


Figure 3.23 : new bogus term print screen

After pressing the "New" button from Figure 3.22, Figure 3.23 appears.

The example shows an unsuccessful attempt to add a new bogus term. One of the requirements is indeed to forbid the addition of a subset of an already existing term.

Modifying a term

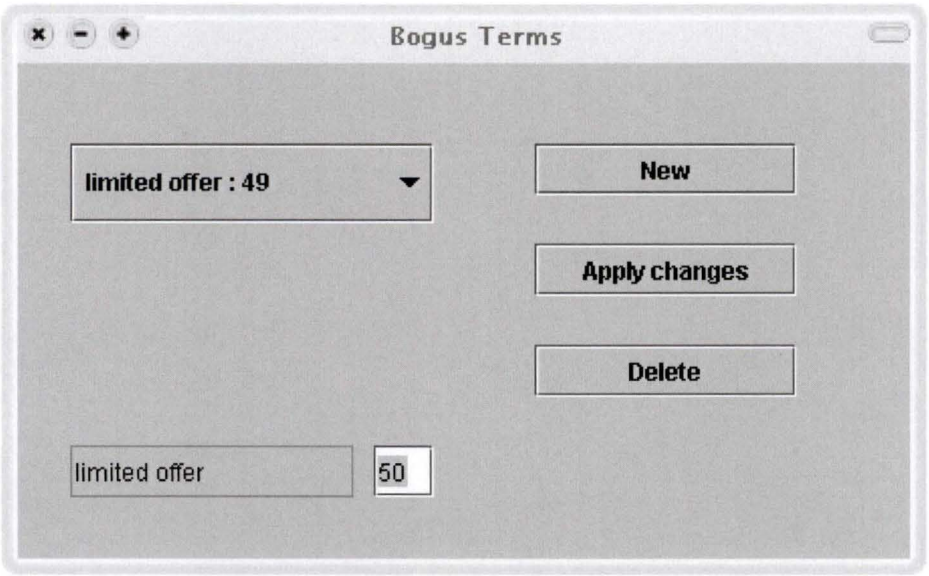


Figure 3.24 : bogus term list print screen

The weight of a term can be easily changed at every moment by selecting it from the list, update its weight and confirm via the "apply changes" button, as shown on Figure 3.24.

Conclusion

In the three chapters of this thesis, we have shown how web information, text mining techniques, and web mining techniques are useful to a chatterbot.
We have explained the global goals and techniques offered by text mining, and pointed out some of them which can be used by a chatterbot to speak in a coherent way.
For this purpose, we divided text mining into two different stages; namely a preparation phase and a processing phase.
We saw that the preparation step includes stop-word removal and stemming to increase effectiveness and that the bag of word representation makes computations easier.
We then observed that some techniques from the processing phase (feature selection from multiple or single documents, similarity measures and clustering), are useful to a chatterbot, whereas others (like classification and summarization) could sometimes be avoided.
We also saw that WordNet is a tool that allows future evolution to a chatterbot using semantics.

We have given a complete progress which can make the bot aware of current events and thus allow the bot to speak in an actualized way. We have seen that this goal can be achieved by two crawlers (i.e. information retrievals), that extract the news from some web sites; namely a manual crawler and a generic crawler.
To use a manual crawler, we have seen that we first have to analyse the structure of a specific web page to discover its layout, before building a template which will be used to extract the information wanted.
We saw that the generic crawler starts to make hyperlink analyzes of a website to "prune" the best links before analyzing the content of those links in order to eliminate the banners advertisements, navigation bars, copyright notices, decoration pictures, etc. and determine the actual informative content block of a page.

Finally, we explained our contribution through the development of a concrete application of some web and text mining techniques within the B.A.D. bot, which adds the particularity to detect people giving advice about stock exchange in a public chatroom. We covered the different types of knowledge owned by our chatterbot, as well as a way to use them within a dialogue. We described the different text mining techniques that were

implemented around news articles as well as the way valuable knowledge is retrieved from internet and managed within the program.

Bibliography

[Bar-Yossef Z. and Rajagopalan S. 2002] "Template detection via Data Mining and its applications" *WWW2002*, Honolulu, Hawaii, USA, May 7-11, 2002

[Donaldson 2001] "Internet Fraud: How to Avoid Internet Investment Scams", http://www.sec.gov/investor/pubs/cyberfraud.htm (Last updated November 15, 2001) (Date of access 15/11/2003)

[Eikvil L. 1999] "Information extraction from the world wide web – a survey", *Report No. 945, ISBN 82-589-02429-0*, July 1999

[Ester M., Kriegel H.P., and Schubert M. 2002] "Web Site Mining: A new way to spot Competitors, Customers and Suppliers in the World Wide Web" *In Proc. of SIGKDD02* Edmonton, Alberta, Canada

[Fayyad U., Shapiro G., and Smyth P. 1996] "From Data Mining to Knowledge Discovery: An Overview", *Advances in Knowledge Discovery and Data Mining, AAAI Press / The MIT Press*, Menlo Park, CA, pp.1-34, 1996

[Hamilton H., Gurak E., Findlater L., and Olive W. 2002] http://www2.cs.uregina.ca/~hamilton/courses/831/notes/kdd/1_kdd.html (Last updated February 7, 2002) (Date of access 15/11/2003)

[Hearst M.A. 1997] "Text Data Mining : Issues, Techniques, and the Relationship to Information Access", *Presentation for UW/MS Workshop on Data Mining,* July 1997

[Hearst M.A. 1999] "Untangling text data mining", In *Proceedings of the Association of Computational Linguistics conference,* June 1999

[Hearst M.A. 2003] http://www.sims.berkeley.edu/~hearst/text-mining.html (Last updated October 17, 2003) (Date of access 11/03/2004)

[Hidalgo J. 2002] http://www.esi.uem.es/~jmgomez/tutorials/ecmlpkdd02/ (Last updated July, 26th 2002) (Date of access 01/03/2004)

[Hsinchun C., Smith T., Larsgaard M., Hill L., Ramsey M. 1997] "A Geographic Knowledge Representation System for Multimedia Geospatial Retrieval and Analysis", *International Journal of Digital Libraries*, 1997

[Kao H.-Y., Ho J.-M., and Chen M. S. 2003] "Clustering for web information Hierarchy Mining" *Web Intelligence 2003*, pages 698-701

[Katz B. 1997] "From Sentence Processing to Information Access on the World Wide Web" *AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, Stanford University, Stanford, CA, 1997

[Katz B. 2003] "Integrating Web-based and Corpus-based Techniques for Question Answering" *In Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, Gaithersburg, Maryland, November 2003

[Kosala R., and Blockeel H. 2000] "Web mining research: A survey" *SIGKDD00*, Volume 2, Issue 1, July 2000

[Lerman K., Knoblock C. and Minton S. 2001] "Automatic Data Extraction from Lists and Tables in Web Sources", *Automatic Text Extraction and Mining Workshop (ATEM-01), International Joint Conference on Artificial Intelligence,* Seattle, August 2001

[Loupy C. 2001] "L'apport de connaissances linguistiques en recherche documentaire", *actes de TALN'2001, Tome 2, pp. 129-143,* Tours, France, 2001

[Mani I., Hahn U. 2000] "The Challenges of Automatic Summarization", *IEEE Computer 33(11), pp. 29-39*

[Mani I., Maybury M.T. 2001] "Advances in Automatic Text Summarization" *Cambridge, Massachusetts: MIT Press*

[Matsuo Y., Ishizuka M. 2003] "Keyword Extraction from a single document using word Co-occurrence statistical information", *FLAIRS Conference 2003, pp. 392-396*

[Méndez-Torreblanca A., Montes-y-Gómez M. and López-López A. 2002] "A Trend Discovery System for Dynamic Web Content Mining", *Proc. Of the XI International Conference on Computing CIC-2002*, Mexico City, Mexico, November 2002

[Miller G. 1998] "WordNet An Electronic Lexical Database", *Edited by Christiane Fellbaum, ISBN 0-262-06197-X, 423 pp.,* May 1998

[Murray B. H. and Moore A. 2000] "Sizing the Internet" *A white paper, Cyveillance,* July 2000

[Porter M. 2001] http://snowball.tartarus.org/texts/introduction.html (Date of access 20/03/2004)

[Porter M. 2002] http://snowball.tartarus.org/english/stop.txt (Last update, September 2002) (Date of access 26/03/2004)

[Salton G. 1988] "Term-Weighting Approaches in Automatic Text Retrieval", *Information Processing & Management, Vol. 24, N. 5, pp. 513-523*

[Salton G. 1989] "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer", *Addison-Wesley*

[Shian-Hua L. and Jan-Ming H. 2002] "Discovering Informative Content Blocks from Web Documents", *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, July 23-26, 2002

[Tian Y., Huang T., and Gao W. 2003] "A web site mining algorithm using the multiscale tree representation model" *Proceedings of the Fifth WEBKDD workshop: Webmining as a Premise to Effective and Intelligent Web Applications (WEBKDD'2003), in conjuction with ACM SIGKDD conference*, pages 83-92, Washington, DC, August 27, 2003

[Van Alstyne M., Brynjolfsson E. 1996] "Electronic Communities: Global Village or Cyberbalkans?", *Best Paper on the conference theme at the 17th International Conference on Information Systems*, Cleveland, OH, Dec. 1996

[Weizenbaum J. 1966] "ELIZA - A Computer Program For the Study of Natural Language Communication Between Man and Machine" *Communications of the ACM*, Volume 9, Number 1, pages 36-35, January 1966

[Xu J., Huang Y., and Madley G. 2003] "A research support system framework for web data mining", *WSS03 Applications, Products and Services of Web-based Support Systems*, pages 37-43.

[Yi L., Liu B., and Li X. 2003] "Eliminating Noisy Information in Web Pages for Data Mining", *SIGKDD .03*, Washington, DC, USA, August 24-27, 2003

[Zalewski M. 2003] http://lcamtif.coredump.cx/catty.txt (Date of access 24/05/04)

Annexe

1. How to use the program

The first thing to do is to launch the batch file named *Chatterbot.bat*. Then a fictive chat room interface appears. This interface represents the Stock Exchange's chat room in which you can write some sentences in order to test and simulate the "bad adviser" person.
In order to do that, you have to write, first, a name followed by a space and ":" (example: "Bob :"). The name of a person has to be different each time you want to test the program (or you can delete it in oracle Smarts database). Then, a sentence can be written but only sentences with "advice(s)" will be detected. When it is done, another frame named "private chat" appears but this frame is only there to see the Agent to "bad adviser" conversation. In fact, there will be as much "private chat" frames as there will be detected "bad advisers".

2. Parameters

- Double clusterlimit = 0.4;

The minimum clusterweight between two words in order to be taken into account in the computations. Depending on the situations, 03 to 1.2 may be used.

- Boolean assymetric_clustering = true;

with asymmetric clustering, f(word1,word2) <> f(word2,word1)
if this parameter is false:
g(word1,word2) = (f(word1,word2)+f(word2,word1))/2

- String explore_cooc = "DEPENDANT"; // or INDEPENDENT

in an indepentant exploration of a sentence, each word is taken as a separate entity and the final 2 words that are chosen are simply the 2 that achieve the highest cluster weight together.

In a dependant exploration, the sentence is seen as an entity. This method is more complex but gives more precise results.

- int headlineprecision = 2;

Defines the minimum amount of times that 2 words have to be present in the same news headline.

- String type_keybord = "QWERTY"; // or AZERTY

Defines the type of keybord the chatterbot "uses" in order to introduce typing errors accordingly

- int maxTimesWithBot = 10;

The maximum amount of times the bot can speak to the same person.
Note that the bot never speaks to the same person twice the same day.

- int minWeightToLaunch = 25;

The minimum bogusweight that is necessary before the bot begins a conversation.

3. DataBases

Package: Acces

Classes: Bd, Bd2

The database access methods are centralised in one class per DataBase, where the concurrent accesses are treated. This centralisation allows the programmers to delegate the treatment of exceptions to these classes.

The program works on two separate databases in order to avoid unnecessary waiting times if a thread accesses a table from the database that is not even linked to an already blocked one.

4. Detail design

4.1 Bogus Detection

Vocabulary

Bogus advice: false advice on buying shares in order to manipulate prices and persons
Share: financial part of a company
Sharecode: each share is identified by a 3, 4 or 5 letter code

Bogus Detection

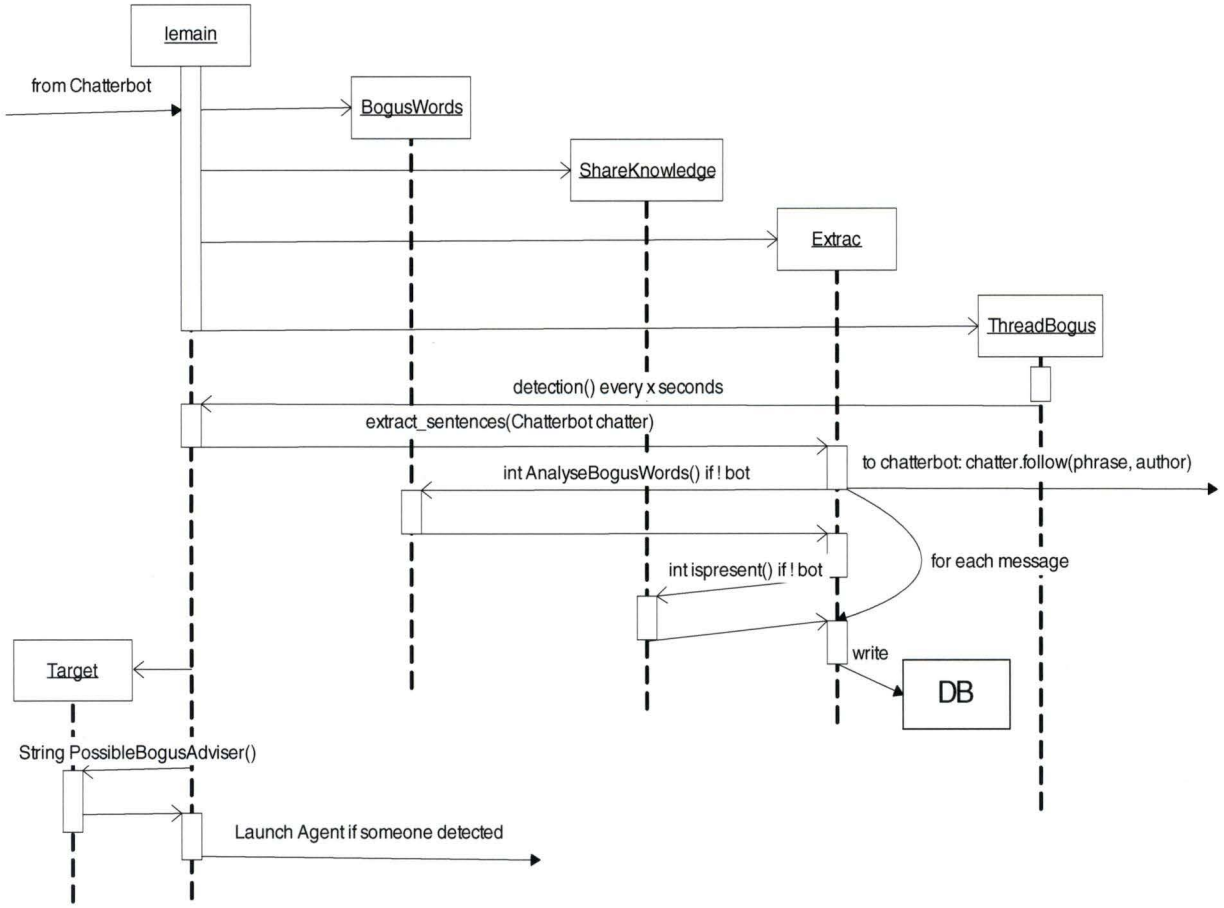
Package : Bogus & General
Classes : Extrac, Lemain, ShareKnowledge, Boguswords, StartBug, General (followFromSmarts)

As opposed to the co-occurrence analysis which is a more complex part (point 5), the detection has been deliberately kept simple and fully understandable for the user of Bad Bots. The purpose is that the person knows and controls who is picked up by our program. The detection of a bogus adviser is based on keywords chosen by the user as well as a "stay in topic" detector. This means that a person speaking about any share will be more easily chosen than a person speaking about whatever else.

Choosing the person the talk to

Package: Bogus
Class: Target

Once a person on the chat has been identified as (perhaps) giving away bogus advice, an agent has to be launched to talk to him. The agent will only be launched if the detected person is not an agent itself and if the person talked within the last 5 minutes. The agent is launched if the bot didn't speak to him during the present day and not more then a maximum amount of times in total. (see Parameters)



4.2 Bogus Adaptor

Application : bogusAdapt

Classes : AccesDB, bogusFrame, newBogusFrame, General

Separated easy to use application that allows the user to add/modify/delete the keywords to be detected as being an indication for bogus as well as their weight (level of importance in the detection).

Keywords are ordered alphabetically and the user can easily keep an eye on all modifications done within the session.

There is an integrated control to avoid doublets and subsequence of words.

4.3 Agent creation

System design

If the avatars are to be "believable" then their design must have a unifying conceptual basis. This is provided here by the agent's *character*. The first decision that is made when an agent is created is to select its character using a semi-random process that ensures that multiple instances of the agent in close virtual proximity have identifiably different characters.

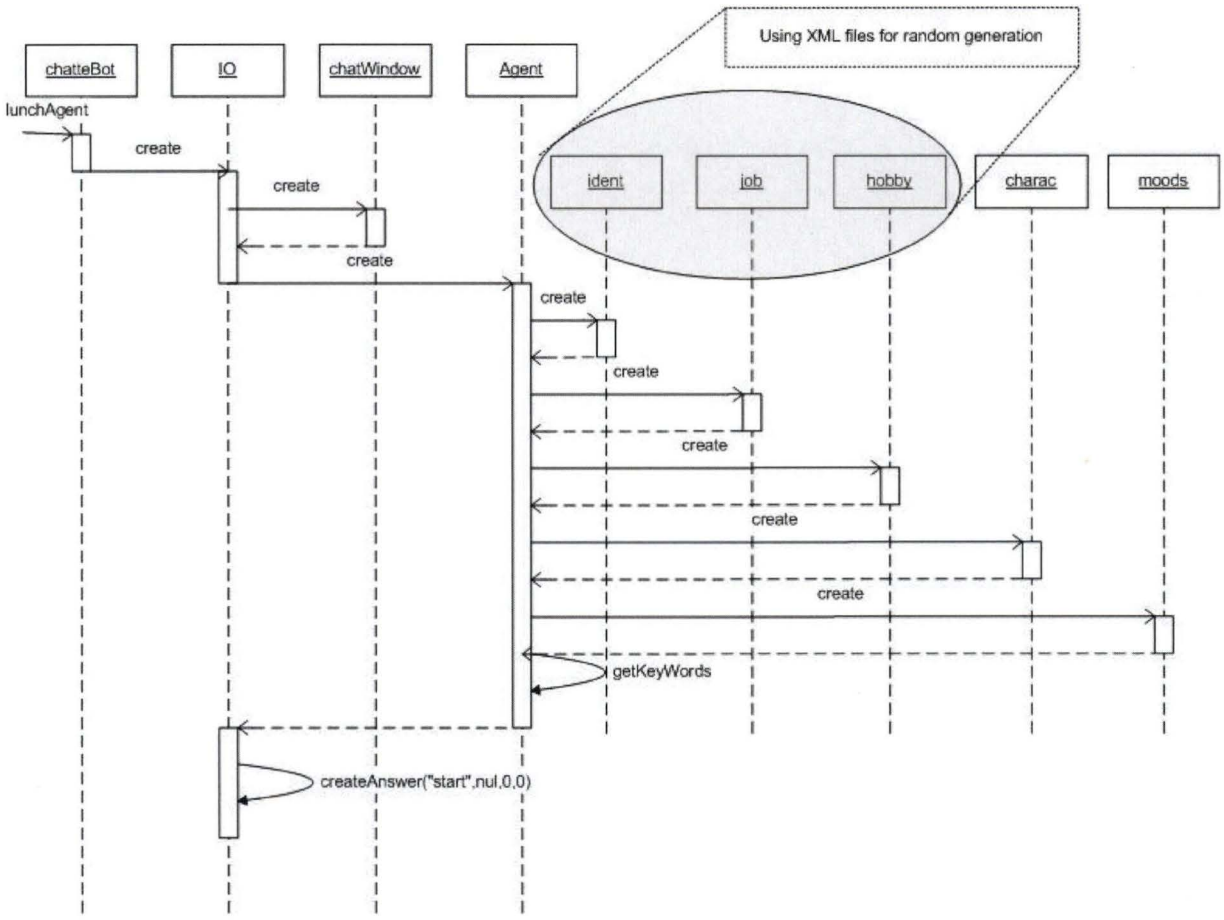
The dimensions of character that we have selected are intended specifically for a finance-based environment.

- Politeness means the use of polite words, phrases and forms
- Dynamism is the tendency to react rapidly, succinctly and vigorously
- Optimism here means a tendency to use up-beat phrases and the tendency to not use negations
- Self-confidence here means the tendency to respond with declarative statements rather than tentative propositions or questions.

The selection on an agent's character does not alone determine its behavior. Each agent's behavior is further determined by its moods that vary slowly but constantly. This is intended to ensure that repeated interactions with the same agent have some degree of novelty. The dimensions of moods that we have identified are:

- Happiness
- Sympathy

Implementation design



4.4 Answer creation

System design

The generated response has to be consistent with the personality and the mood of the character and if possible, the answer will contain some news from Internet.

In order to achieve that, XML technology is used for response's creation. In fact, tree XML files are used, listWords.xml, ListAnsWithoutOnt.xml, ListAnswerOnt.xml.

First, the sentence which comes from the chat room is parsed in order to find a topic for the answer's creation. There are five different topics, Shares, Co-occurrence, Weather, Financial News and nope. Then we analyse the sentence again to be sure that it is not a "key" sentence like "what's your name" or "what's your job", etc. If not, then we have two possibilities :

either the found topic is one of the fourth first ones and then we are going to use the ListAnswerOnt file or the return topic is "nope" and then we are going to use the ListAnsWithoutOnt file.

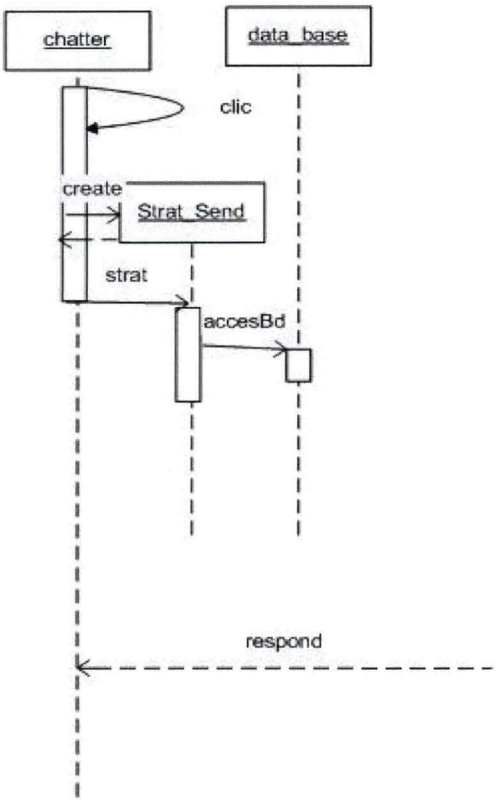
But first, ListWords is use to find a theme corresponding to a given list of words (topic).

The obtained thema allows now to choose a focus node (in one of the too others XML files) having the same attribute theme. The attribute mark-up from the markupPhrases node represents, for the first figure, the politeness, and for the second, the optimism of an Agent. It allows then to select an output sentence corresponding to the personality of an Agent.

A way to look more human is to introduce random keyboard mistakes into the chatterbot's replies. The choice is left to the user of our program to have the chatterbot using a querty or azerty keyboard, as the mistakes are different on both types of keyboards.

Implementation design

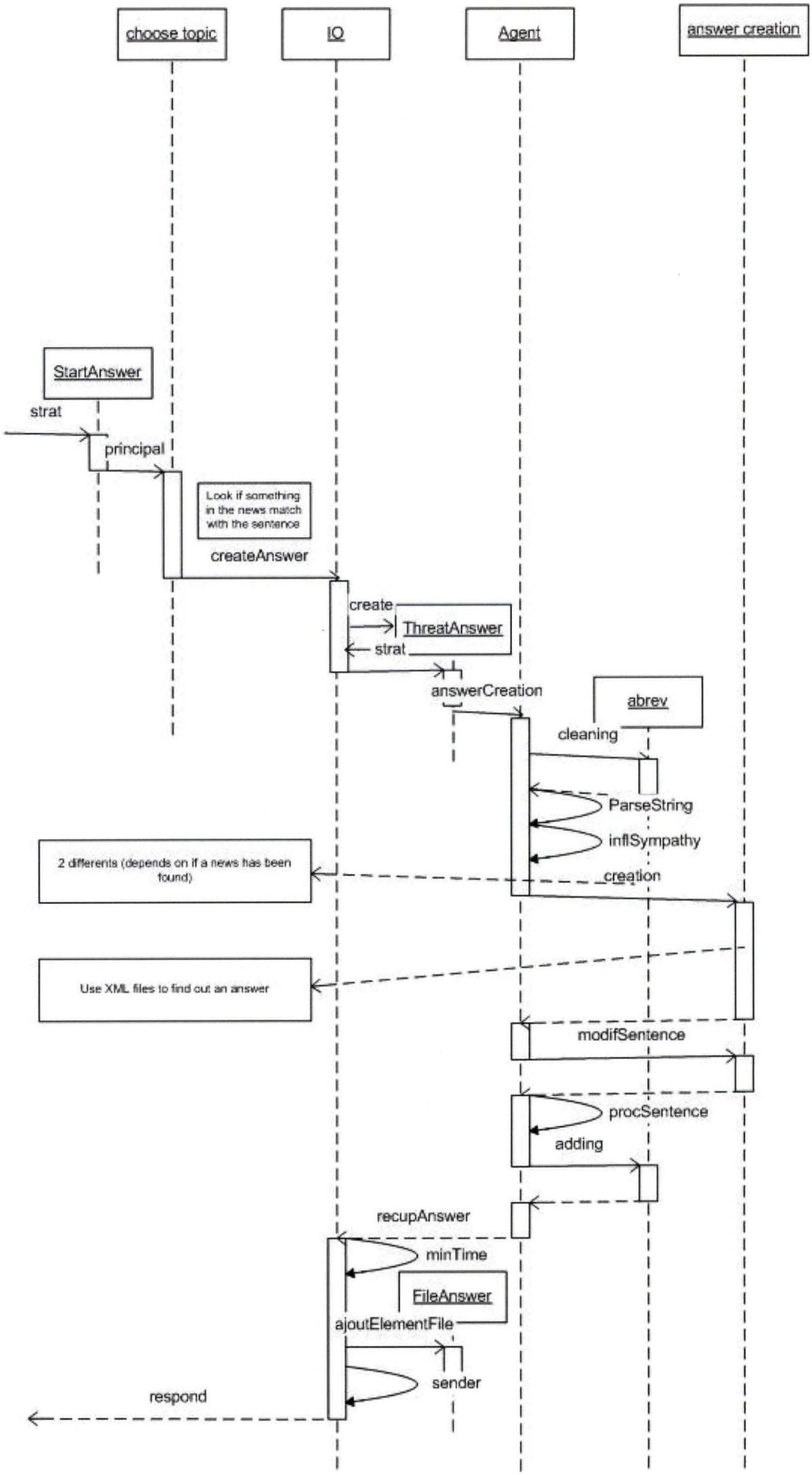
First step: the "fictive" chat room launching.



Second step: the method "followFromSmart" which is called in the bogus detection module launch every **x** second(s) a thread named StartAnswer.

But in this case, we have too possibilities:

1) If there are two (or more) StartAnswer thread launched before one has finished its execution, then the othes threads are placed in a waiting list named "FileAnswer"
2) Else, the method "respond" is launched directly.


choose topic
IO
Agent
answer creation
StartAnswer
strat
principal
Look if something in the news match with the sentence
createAnswer
create
ThreatAnswer
strat
answerCreation
abrev
cleaning
ParseString
inflSympathy
2 differents (depends on if a news has been found)
creation
Use XML files to find out an answer
modifSentence
procSentence
adding
recupAnswer
minTime
FileAnswer
ajoutElementFile
sender
respond

4.5 Co-occurrence

Vocabulary

Clustering: grouping of objects based on similarity, dissimilarity or any other property
Co-occurrence: method based on occurrences of words in the same texts

Package: Co-occurrence
Class : Cooc

The proposed pre-processing phase still has to be linked to this module.
The initial model we chose to use is a clustering method that calculates a weight between words representing their co-occurrence level in a series of texts. Unfortunately, in spite of very nice suggestions obtained with the method, we had to adapt it to reply with a particular newsheadline (full sentence), which actually gives results that are less impressive, but still accurate if the program is well parametered. Each phrase a user says can be analyzed by the bot in a dependant or independent way.
A particular attention has been brought to optimization, both in memory and real-time performance.
The clustering method is asymmetric; this means that the weight between word1 and word2 is not the same as between word2 and word1.
The Cooc method works in three phases.
1. A few times a day: building a datastructure representing the content of the texts as well as mathematical computations. Some concessions have been made in this phase to improve the speed of the next real-time phases. This happens every time news has successfully been fetched from the internet. (x times a day, nothing is computed if no news has been retrieved)
If concepts are not used, this part has been brought to 10 seconds in its actual version thanks to many optimization techniques. (for 100 texts and 15000 words on a Celeron 1000 with 256 MB Ram).
2. For every intervention of the chattermate: word to word clustering
3. Choice of the best newsheadline

4.6 Purgatoire

Package : Purger
Class : Purgatoire

The chatterbot has to be able to acquire new knowledge and to stay up-to-date with its environment but it would be silly if it talked suddenly about a breaking news that happened 3 weeks ago …
The Purger is called once a day and has a forgetting curve that is not linear, as the bot wipes its knowledge of, based on a probability and the age of the knowledge.
Weather and shares don't have to be purged, because the content is updated and not added.

4.7 The concept network

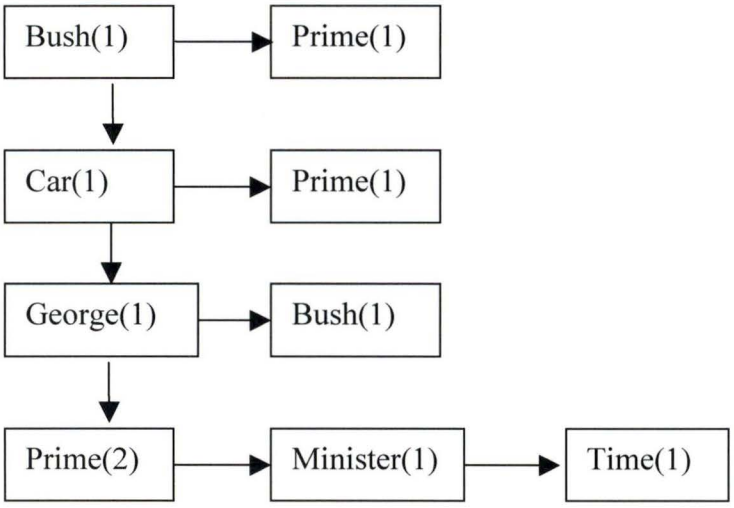
What is a concept

A concept can be a one-, two-, three, or more-word phrase. If a word is almost every time followed by another one, we can say that it is a concept.
Prime minister, hunger strike, George W Bush, foreign affaires, … are examples of concepts.

Representation of the concepts

Here is the exact structure of the linked list with this two sentences:

- George Bush Prime Minister
- Car Prime time



The first column (le left one) represents what we call "words" and the other column (here there are only 2 others) represent what we call "links". The number between brackets has a different signification for the words and for the links. For the words it represents the total number of occurrences of this word. For the links it represents the number of time that the link appears just behind the word of row. For example "Prime" appears two times in all the texts, and "Time" appears one time behind "Prime".

4.8 The News Fetcher

The person that we are going to interact with is the person that talks a lot about finance and particularly, Australian shares.
Where will we take the information from?

Shares

The link to follow is from the official Australian Stock exchange website, the access to the information is quiet complicated because there is no direct link to each share.
The webite uses a combobox and we have to select all the shares one by one.

- http://www.asx.com.au/asx/markets/EquitySearchPage.jsp?template=sfl1000&issuername=&x=7&y=8

Financial News

- Australia & NZ Finance (http://au.dailynews.yahoo.com/finance/reutersfinance/)
  - Headline
  - Location
  - Month (3 letters) comma day (digits)

World News

ABC news online seems the best website, the information is structured and complete. We save the headlines together with the date of the news and the full story associated. This information are also used to build the co-occurrences structure and the concept network.

- ABC news online http://www.abc.net.au/news/justin/default.htm

Weather

Also taken from ABC, a 4 day weather forecast with a very short comment for each day. Information is available for most Australian cities.

- Australia Centres Weather Forecast (http://www.abc.net.au/news/australia/weather/default.htm)

4.9. Launching an Agent

Once a person on the chat has been identified as (perhaps) giving away bogus advice, an agent has to be launched to talk to him. The agent will only be launched if the detected person is not an agent itself and if the person talked within the past 5 minutes. These 5 minutes are a way to prevent an agent to be launched on a user that went away a long time ago in case of a system crash.

The launching of an agent depends on two parameters chosen by the user:

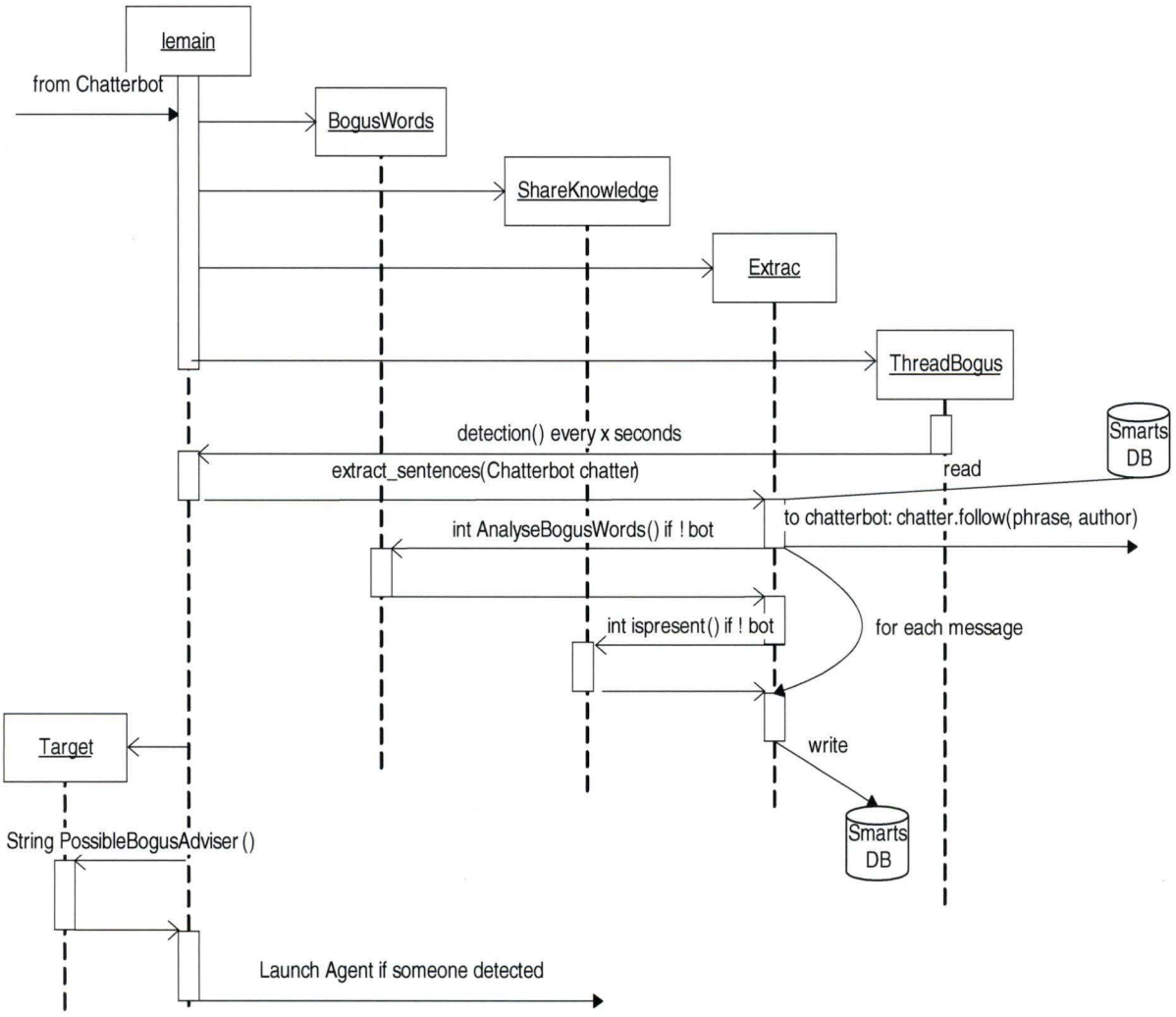
maxTimesWithBot = 10

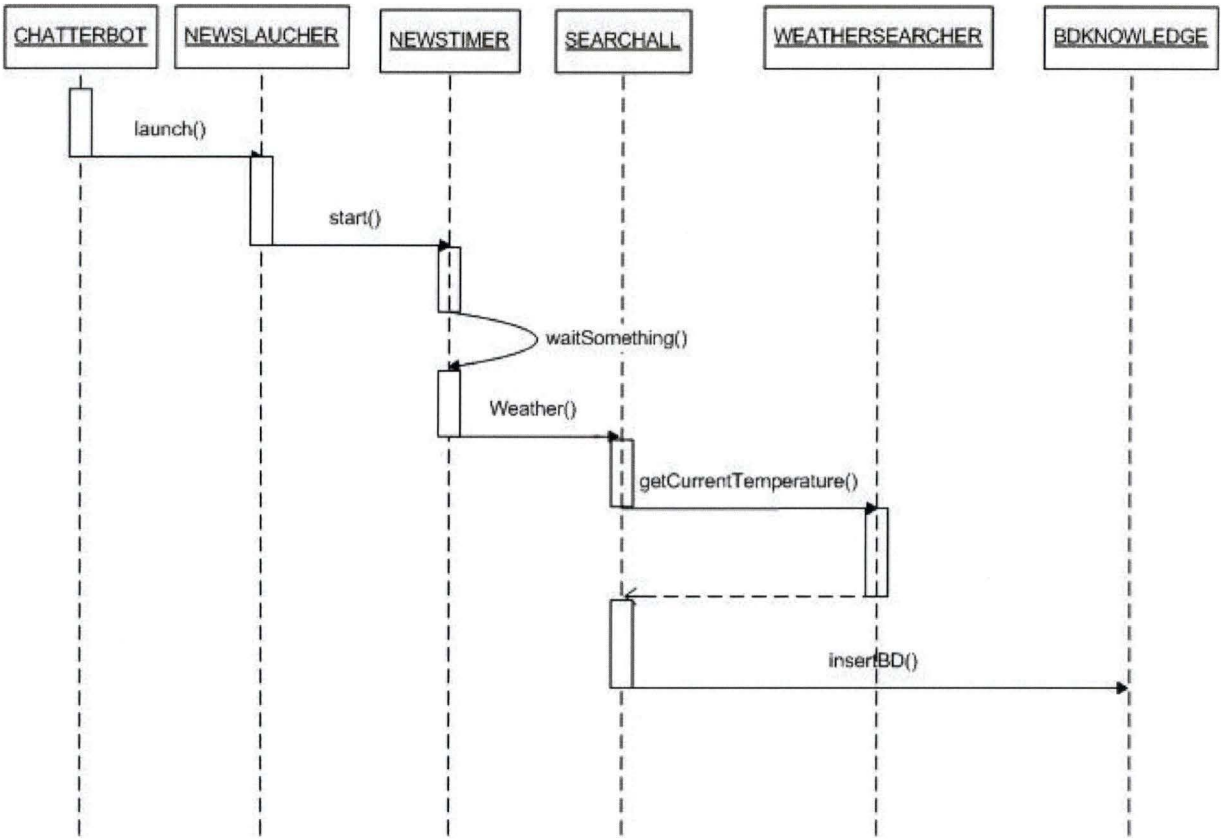
int minWeightToLaunch = 90

The parameters speak for themselves.

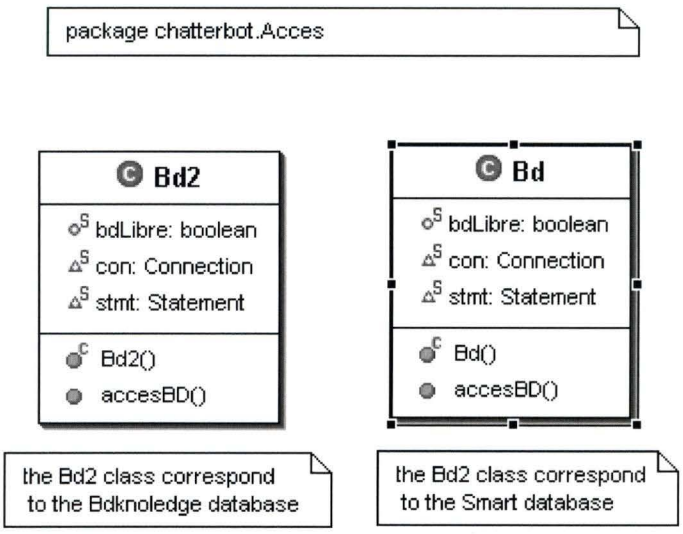
Note that we have chosen to forbid an agent to be launched on a particular person if the bot already talked to him during the present day.

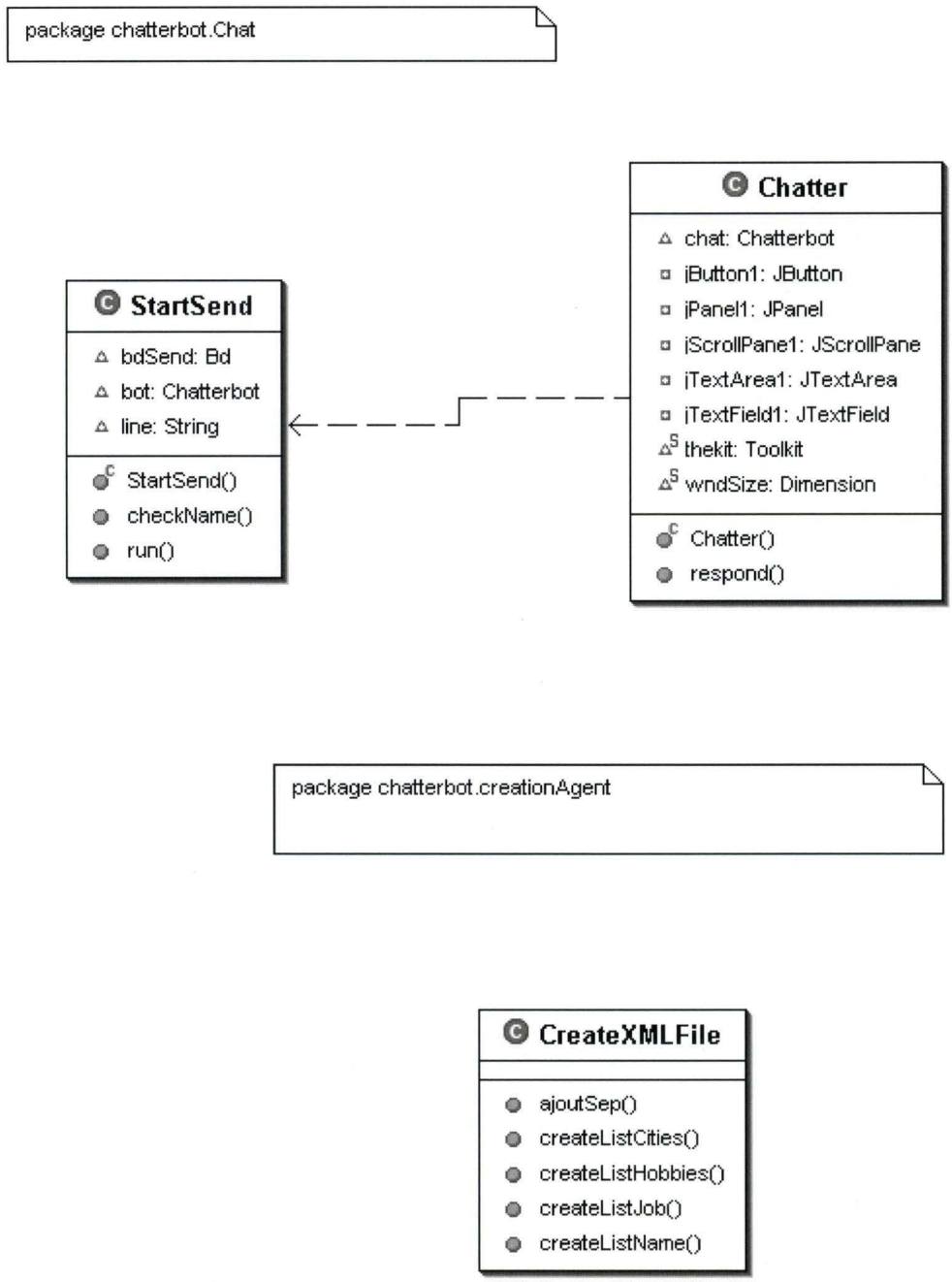
The next figure shows the sequence of the detection followed by the launching of an agent.

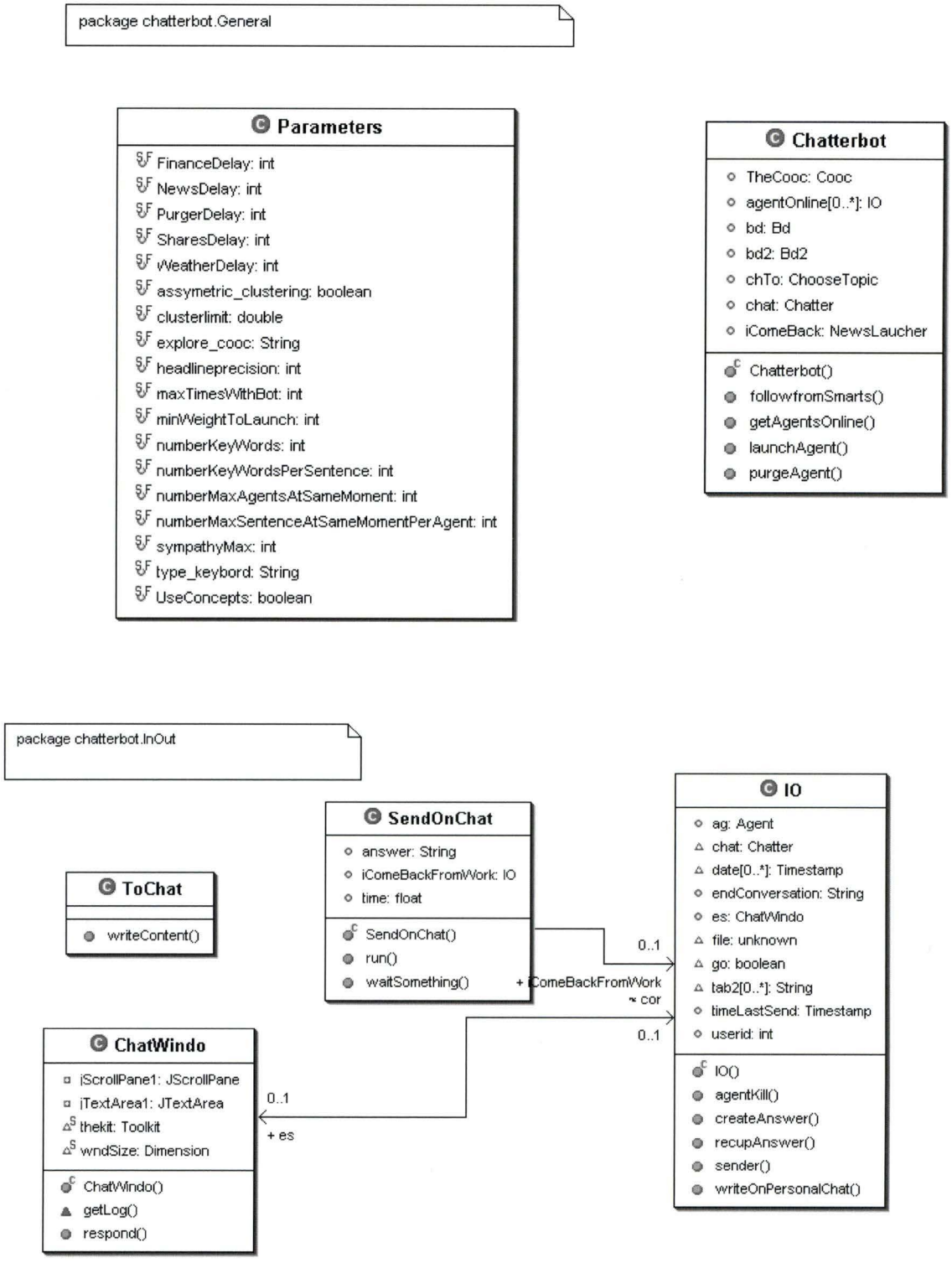


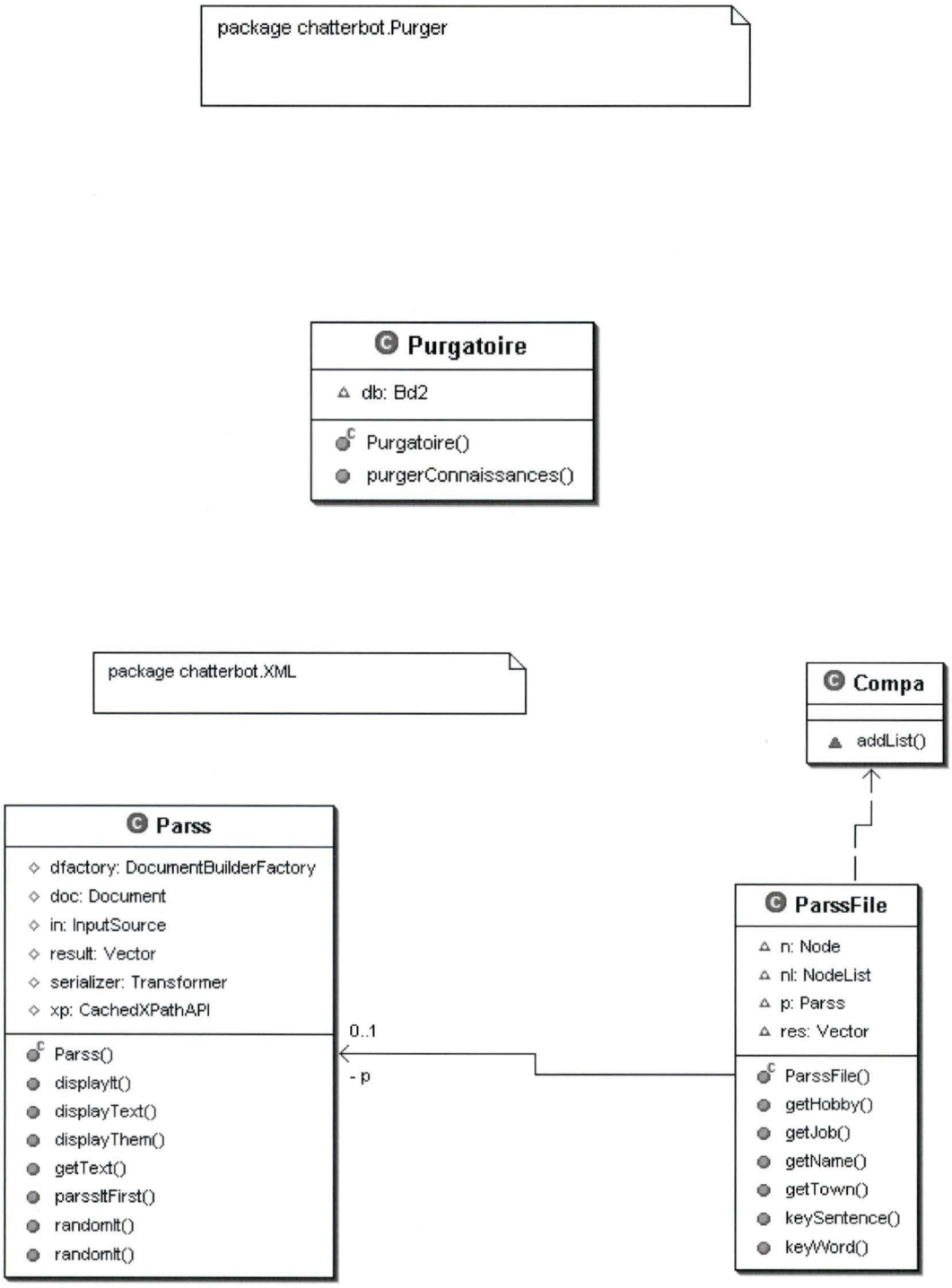


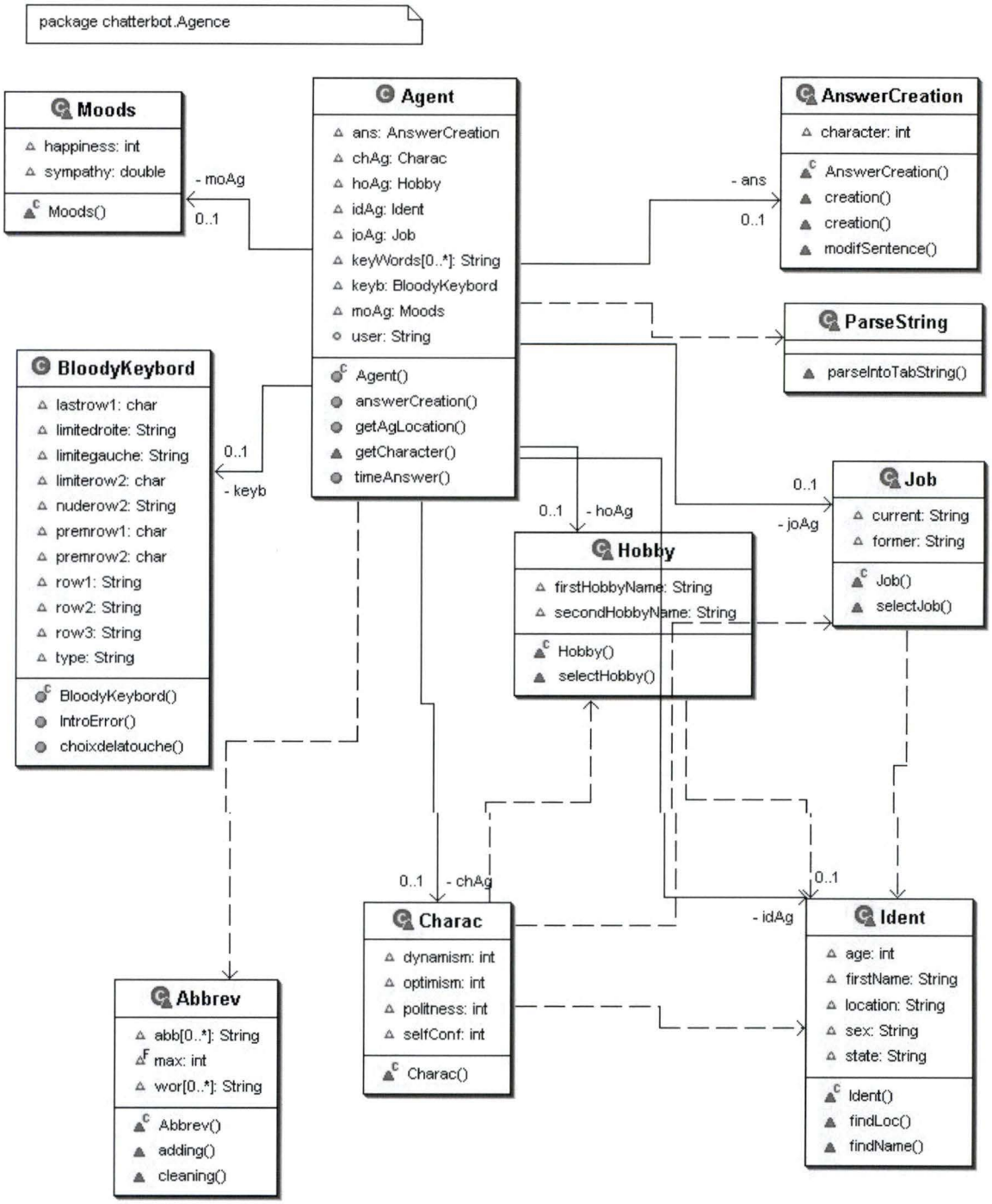
5. *Class Diagrams*

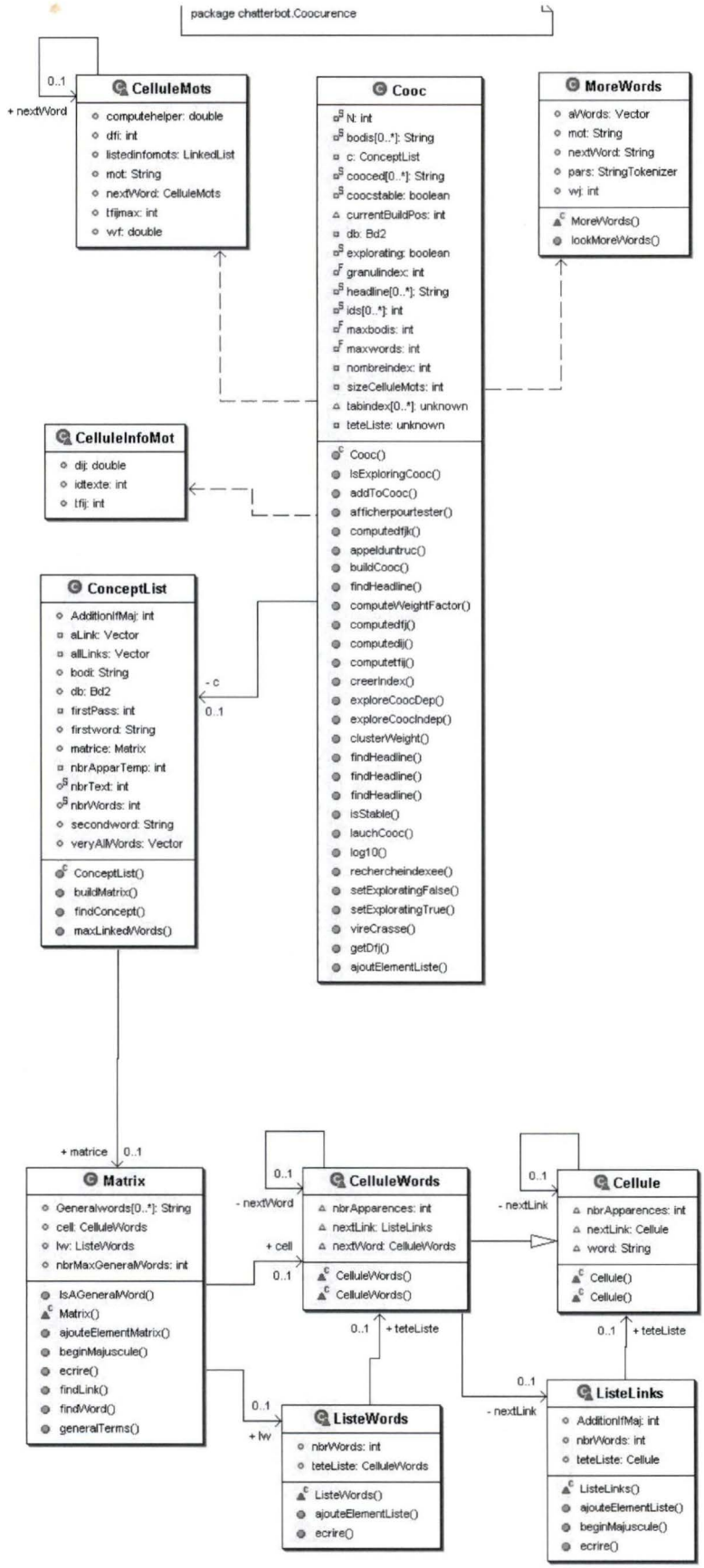


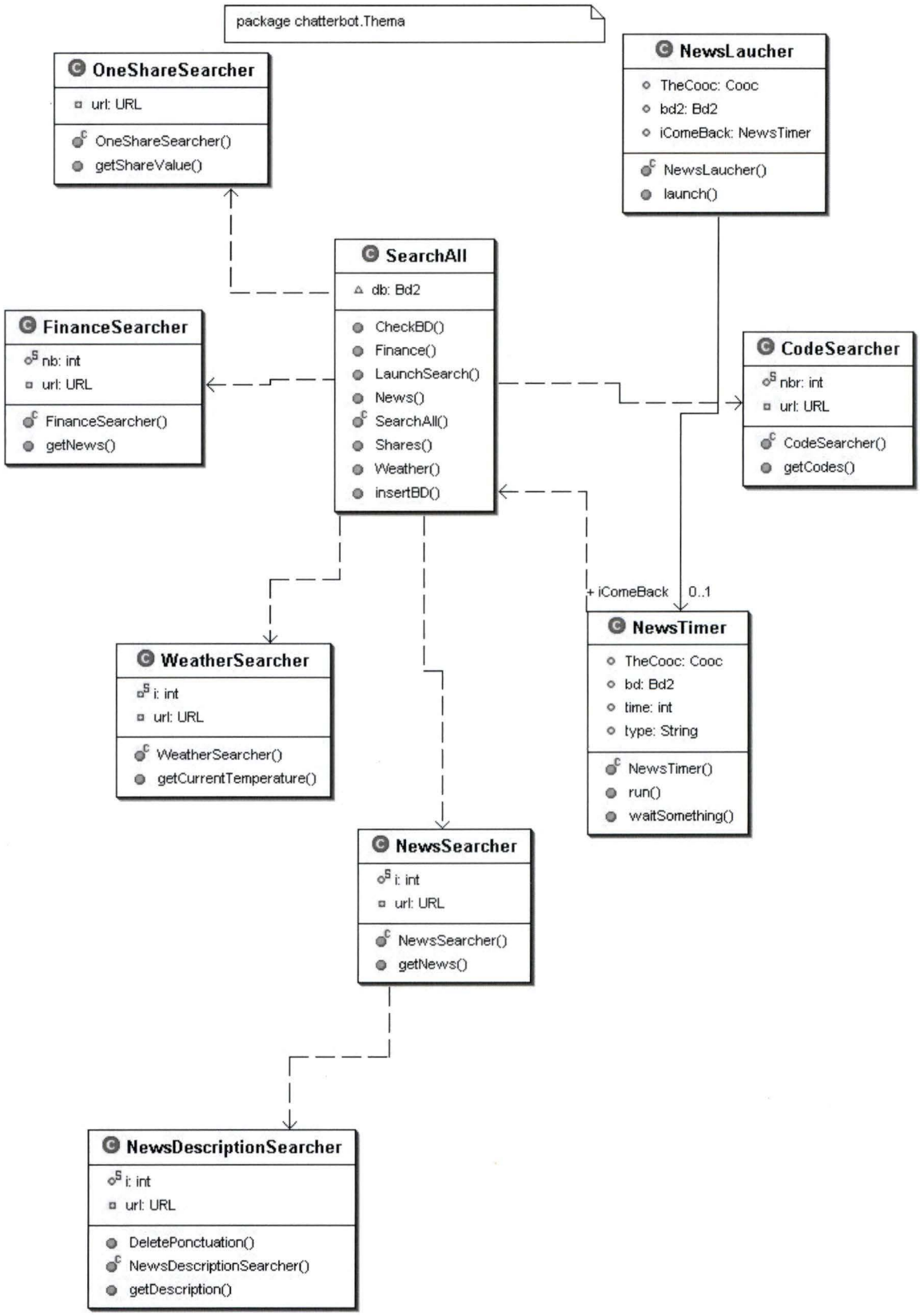

package chatterbot.Chat
Chatter
chat: Chatterbot
jButton1: JButton
jPanel1: JPanel
jScrollPane1: JScrollPane
jTextArea1: JTextArea
jTextField1: JTextField
thekit: Toolkit
wndSize: Dimension
Chatter()
respond()
StartSend
bdSend: Bd
bot: Chatterbot
line: String
StartSend()
checkName()
run()
package chatterbot.creationAgent
CreateXMLFile
ajoutSep()
createListCities()
createListHobbies()
createListJob()
createListName()

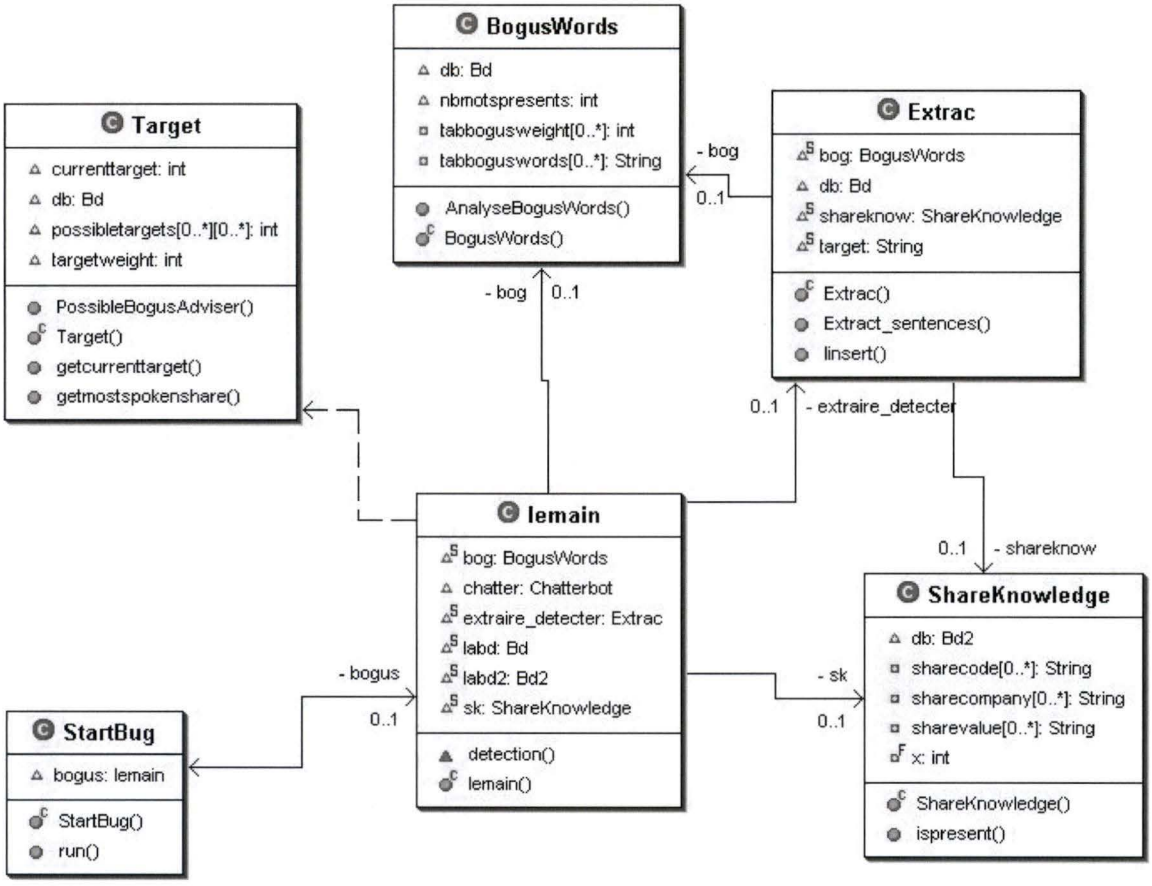

package chatterbot.General
Parameters
FinanceDelay: int
NewsDelay: int
PurgerDelay: int
SharesDelay: int
WeatherDelay: int
assymetric_clustering: boolean
clusterlimit: double
explore_cooc: String
headlineprecision: int
maxTimesWithBot: int
minWeightToLaunch: int
numberKeyWords: int
numberKeyWordsPerSentence: int
numberMaxAgentsAtSameMoment: int
numberMaxSentenceAtSameMomentPerAgent: int
sympathyMax: int
type_keybord: String
UseConcepts: boolean
Chatterbot
TheCooc: Cooc
agentOnline[0..*]: IO
bd: Bd
bd2: Bd2
chTo: ChooseTopic
chat: Chatter
iComeBack: NewsLaucher
Chatterbot()
followfromSmarts()
getAgentsOnline()
launchAgent()
purgeAgent()
package chatterbot.InOut
ToChat
writeContent()
SendOnChat
answer: String
iComeBackFromWork: IO
time: float
SendOnChat()
run()
waitSomething()
0..1
+ ComeBackFromWork
~ cor
0..1
IO
ag: Agent
chat: Chatter
date[0..*]: Timestamp
endConversation: String
es: ChatWindo
file: unknown
go: boolean
tab2[0..*]: String
timeLastSend: Timestamp
userid: int
IO()
agentKill()
createAnswer()
recupAnswer()
sender()
writeOnPersonalChat()
ChatWindo
jScrollPane1: JScrollPane
jTextArea1: JTextArea
thekit: Toolkit
wndSize: Dimension
ChatWindo()
getLog()
respond()
0..1
+ es


package chatterbot.Purger
Purgatoire
db: Bd2
Purgatoire()
purgerConnaissances()
package chatterbot.XML
Compa
addList()
Parss
dfactory: DocumentBuilderFactory
doc: Document
in: InputSource
result: Vector
serializer: Transformer
xp: CachedXPathAPI
Parss()
displayIt()
displayText()
displayThem()
getText()
parssItFirst()
randomIt()
randomIt()
0..1
- p
ParssFile
n: Node
nl: NodeList
p: Parss
res: Vector
ParssFile()
getHobby()
getJob()
getName()
getTown()
keySentence()
keyWord()


package chatterbot.Agence
Moods
happiness: int
sympathy: double
Moods()
- moAg
0..1
Agent
ans: AnswerCreation
chAg: Charac
hoAg: Hobby
idAg: Ident
joAg: Job
keyWords[0..*]: String
keyb: BloodyKeybord
moAg: Moods
user: String
Agent()
answerCreation()
getAgLocation()
getCharacter()
timeAnswer()
- ans
0..1
AnswerCreation
character: int
AnswerCreation()
creation()
creation()
modifSentence()
ParseString
parseIntoTabString()
BloodyKeybord
lastrow1: char
limitedroite: String
limitegauche: String
limiterow2: char
nuderow2: String
premrow1: char
premrow2: char
row1: String
row2: String
row3: String
type: String
BloodyKeybord()
IntroError()
choixdelatouche()
0..1
- keyb
0..1
- hoAg
Hobby
firstHobbyName: String
secondHobbyName: String
Hobby()
selectHobby()
0..1
- joAg
Job
current: String
former: String
Job()
selectJob()
0..1
- chAg
Charac
dynamism: int
optimism: int
politness: int
selfConf: int
Charac()
0..1
- idAg
Ident
age: int
firstName: String
location: String
sex: String
state: String
Ident()
findLoc()
findName()
Abbrev
abb[0..*]: String
max: int
wor[0..*]: String
Abbrev()
adding()
cleaning()


package chatterbot.Coocurence
CelluleMots
computehelper: double
dfi: int
listedinfomots: LinkedList
mot: String
nextWord: CelluleMots
tfijmax: int
wvf: double
0..1
+ nextWord
Cooc
N: int
bodis[0..*]: String
c: ConceptList
cooced[0..*]: String
coocstable: boolean
currentBuildPos: int
db: Bd2
exploratingj: boolean
granulindex: int
headline[0..*]: String
ids[0..*]: int
maxbodis: int
maxwords: int
nombreindex: int
sizeCelluleMots: int
tabindex[0..*]: unknown
teteListe: unknown
Cooc()
IsExploringCooc()
addToCooc()
afficherpourtester()
computedfjk()
appelduntruc()
buildCooc()
findHeadline()
computeWeightFactor()
computedfj()
computedij()
computetfij()
creerIndex()
exploreCoocDep()
exploreCoocIndep()
clusterWeight()
findHeadline()
findHeadline()
findHeadline()
isStable()
lauchCooc()
log10()
rechercheindexee()
setExploratingFalse()
setExploratingTrue()
vireCrasse()
getDfj()
ajoutElementListe()
MoreWords
aWords: Vector
mot: String
nextWord: String
pars: StringTokenizer
wj: int
MoreWords()
lookMoreWords()
CelluleInfoMot
dij: double
idtexte: int
tfij: int
ConceptList
AdditionIfMaj: int
aLink: Vector
allLinks: Vector
bodi: String
db: Bd2
firstPass: int
firstword: String
matrice: Matrix
nbrApparTemp: int
nbrText: int
nbrWords: int
secondword: String
veryAllWords: Vector
ConceptList()
buildMatrix()
findConcept()
maxLinkedWords()
- c
0..1
+ matrice
0..1
Matrix
GeneralWords[0..*]: String
cell: CelluleWords
lw: ListeWords
nbrMaxGeneralWords: int
IsAGeneralWord()
Matrix()
ajouteElementMatrix()
beginMajuscule()
ecrire()
findLink()
findWord()
generalTerms()
0..1
- nextWord
CelluleWords
nbrApparences: int
nextLink: ListeLinks
nextWord: CelluleWords
CelluleWords()
CelluleWords()
+ cell
0..1
0..1
- nextLink
Cellule
nbrApparences: int
nextLink: Cellule
word: String
Cellule()
Cellule()
0..1
+ teteListe
0..1
+ teteListe
0..1
+ lw
ListeWords
nbrWords: int
teteListe: CelluleWords
ListeWords()
ajouteElementListe()
ecrire()
0..1
- nextLink
ListeLinks
AdditionIfMaj: int
nbrWords: int
teteListe: Cellule
ListeLinks()
ajouteElementListe()
beginMajuscule()
ecrire()


package chatterbot.Thema
OneShareSearcher
url: URL
OneShareSearcher()
getShareValue()
NewsLaucher
TheCooc: Cooc
bd2: Bd2
iComeBack: NewsTimer
NewsLaucher()
launch()
SearchAll
db: Bd2
CheckBD()
Finance()
LaunchSearch()
News()
SearchAll()
Shares()
Weather()
insertBD()
FinanceSearcher
nb: int
url: URL
FinanceSearcher()
getNews()
CodeSearcher
nbr: int
url: URL
CodeSearcher()
getCodes()
+ iComeBack
0..1
NewsTimer
TheCooc: Cooc
bd: Bd2
time: int
type: String
NewsTimer()
run()
waitSomething()
WeatherSearcher
i: int
url: URL
WeatherSearcher()
getCurrentTemperature()
NewsSearcher
i: int
url: URL
NewsSearcher()
getNews()
NewsDescriptionSearcher
i: int
url: URL
DeletePonctuation()
NewsDescriptionSearcher()
getDescription()


BogusWords
db: Bd
nbmotspresents: int
tabbogusweight[0..*]: int
tabboguswords[0..*]: String
AnalyseBogusWords()
BogusWords()
Target
currenttarget: int
db: Bd
possibletargets[0..*][0..*]: int
targetweight: int
PossibleBogusAdviser()
Target()
getcurrenttarget()
getmostspokenshare()
Extrac
bog: BogusWords
db: Bd
shareknow: ShareKnowledge
target: String
Extrac()
Extract_sentences()
linsert()
- bog
0..1
- bog
0..1
0..1
- extraire_detecter
lemain
bog: BogusWords
chatter: Chatterbot
extraire_detecter: Extrac
labd: Bd
labd2: Bd2
sk: ShareKnowledge
detection()
lemain()
0..1
- shareknow
ShareKnowledge
db: Bd2
sharecode[0..*]: String
sharecompany[0..*]: String
sharevalue[0..*]: String
x: int
ShareKnowledge()
ispresent()
- bogus
0..1
- sk
0..1
StartBug
bogus: lemain
StartBug()
run()