Thematic Annotation: extracting concepts out of documents

Pierre Andrews
pierre.andrews@a3.epfl.ch
Martin Rajman
martin.rajman@epfl.ch

Technical Report IC/2004/68

School of Computer & Communication Sciences
Swiss Federal Institute of Technology, Lausanne

Artificial Intelligence Laboratory
Institute of Core Computing Science

August 2004

Abstract

Semantic document annotation may be useful for many tasks. In particular, in the framework of the MDM project[1], topical annotation – i.e. the annotation of document segments with tags identifying the topics discussed in the segments – is used to enhance the retrieval of multimodal meeting records. Indeed, with such an annotation, meeting retrieval can integrate topics in the search criteria offered to the users.

Contrarily to standard approaches to topic annotation, the technique used in this work does not centraly rely on some sort of – possibly statistical – keyword extraction. In fact, the proposed annotation algorithm uses a large scale semantic database – the EDR Electronic Dictionary[2] – that provides a concept hierarchy based on hyponym and hypernym relations.This concept hierarchy is used to generate a synthetic representation of the document by aggregating the words present in topically homogeneous document segments into a set of concepts best preserving the document's content.

The identification of the topically homogeneous segments – often called Text Tiling – is performed to ease the computation as the algorithm will work on smaller text fragments. In addition, it is believed to improve the precision of the extraction as it is performed on topically homogeneous segments. For this task, a standard techniques – proposed by [RSA97] – relying on similarity computation based on vector space representations have been implemented. Hence, the main challenge in the project was to create a novel topic identification algorithm, based on the available semantic resource, that produces good results when applied on the automatically generated segments.

This new extraction technique uses an unexplored approach to topic selection. Instead of using semantic similarity measures based on a semantic resource, the later is processed to extract the part of the conceptual hierarchy relevant to the document content. Then this conceptual hierarchy is searched to extract the most relevant set of concepts to represent the topics discussed in the document. Notice that this algorithm is able to extract generic concepts that are not directly present in the document.

The segmentation algorithm was evaluated on the Reuters corpus, composed of 806'791 news items. These items were aggregated to construct a single virtual document where the algorithm had to detect boundaries. These automatically

[1] http://www.issco.unige.ch/projects/im2/mdm/

[2] http://www.iijnet.or.jp/edr/

generated segments were then compared to the initial news items and a metric has been developed to evaluate the accuracy of the algorithm.

The proposed approach for topic extraction was experimentally tested and evaluated on a database of 238 documents corresponding to bibliographic descriptions extracted from the INSPEC database[3]. A novel evaluation metric was designed to take into account the fact that the topics associated with the INSPEC descriptions – taken as the golden truth for the evaluation – were not produced based on the EDR dictionary, and therefore needed to be approximated by the available EDR entries.

Alltogether, the combination of existing document segmentation methods – i.e text tiling – with novel topic identification ones leads to an additional document annotation useful for more robust retrieval.

[3] http://www.iee.org/publish/inspec/

Contents

Chapter 1

Introduction

1.1 Goals and issues

Automatic document summarization is often regarded as a technique to extract a set of sentences from the document. This set of sentences can then be displayed to the user for an easier understanding of a document's themes. However, when used for information retrieval, theme descriptors should be less constraining than full sentences. The user might prefer to describe his query with simple keywords and keyword extraction methods have been developed for this purpose. The problem with such a method is that it's hard to retain the semantics of the whole document in a *few keywords*. Hence, an annotation method offering a tradeoff between summarization – that preserves most of the semantics – and keywords extraction – that are easier to query, but loose a lot of the semantics – has been investigated.

The aim of this project is to provide a list of topics – i.e. a small list of words or compounds each representing a concept– for the indexation of the processed documents. Hence a technique extracting a *limited* list of topics that represents the document subjects has been developed. The main goal is then, not to extract keywords from the documents, but words representing the topics present in this document. These words should preserve more information about the content of the document as they describe concepts and not just words present in this document.

We believe that the use of an ontology can help the for this process, in particular for aggregation of conceptually similar words. A semantic database (like the EDR Electronic Dictionary) provides a dictionary where all words are associated with a gloss and the concepts it can represent. Each of these concepts is also positioned in a Directed Acyclic Graph representing a semantic classification; each concept is attached to one or more super-concepts – having a more general meaning – and zero or more sub-concepts – having a more precise meaning. The EDR database can be used to identify the concepts attached to each word of our document. The document is then represented as a bag of concepts, and the challenge is to use the conceptual structure to deduce the discussed topics.

The EDR database provides about 489'000 concepts in its hierarchy; such a large number implies that, for one word in our document, we will usually have more than one concept associated with it. Aggregating these concepts for a big document can be computationally inefficient as the resulting conceptual DAG will be large as well. For this reason, the direct processing of large documents is not realistic in the perspective of the extraction of relevant topics.

An possible approach to process large documents is to arbitrarily divide them into multiple parts and to extract the topics for each of these parts. However, to keep a certain coherence in the segmentation and to give the topic extraction algorithm a better chance, the segmentation cannot be made randomly. Keeping a homogeneity in the segment's semantics increases the probabilities to extract the more relevant concepts from each part of the document. Good techniques already exist to segment texts in homogeneous blocks and provide simple statistical algorithms that can be implemented without many computational obstacles.

1.2 Segmentation

The Text Tiling method [Hea94] that has been chosen is a simple statistical method (see chapter 5). It uses a bag of words representation of the document to identify segments:

- the document is split in N windows of fixed size,
- a bag of words representation is constructed associating a weight to each token,
- the distance between every two consecutive window is computed,
- thematic shifts are identified by searching the positions in the text where these distances are minimal.

This method has the advantage of requiring little preprocessing and no external knowledge source. Segments are only bound by intra document computations. However, if not tuned correctly this technique can lead to poor results. Initial windows size is critical and varies for each text. User input would be optimal to adjust the algorithm parameters , but is not in the scope of our system. Luckily, to perform the topic extraction, the segments do not have to be exact and a *noisy* segmentation is therefore admissible in our case.

1.3 Topic Extraction

The topic extraction method that has been developed for this project has barely been studied from such a point of view before. Indeed the use of an external knowledge source often translates into additions to the basic bag of words representation produced for the document, or by the computation of semantic similarities (see section 2). In this project, another approach has been chosen:

- for each word of the document, we identify the lemmas from which it can derive,
- a list of leaf concepts is extracted from the semantic database for each lemma,
- starting from the leaf level, the conceptual hierarchy is constructed level by level,
- a set of concepts corresponding a cut[1] in this hierarchy is selected to describe the document.

This method is believed to be more efficient as it does not compute the similarity of each possible pair of concepts but only constructs a subhierarchy of the complete ontology. Preprocessing for this task have been kept simple and only part of speech disambiguation is performed (see section 5.1) before the construction of the DAG. The weak point of the current version of the algorithm is the absence of any Word Sense Disambiguation (see section 5.1.2), as a word in the document can trigger more than one concept in the hierarchy – each of these concepts representing a different word sense.

1.4 Document Structure

This report has been divided in four main parts. Chapter 2 presents the state of the art of the existing techniques that have influenced the developments made during the project. The next Chapter (Chapter 3) presents the main resource used during this project and the constraints that had to be taken into account to conform to the available linguistic resource (in addition, see Section 5.1.1).

The two main parts of the project are presented in Chapters 4 and 5. The first presents the implementation of the Text Tiling techniques, while the second describes our novel topic extraction technique.

Chapters 6 and 8 present the evaluation process and the conclusions that were reached from it. 7 offers some hints on the future work that can be lead in continuation to this project.

[1] see section 5.3.1 for the definition of a cut in a DAG.

Chapter 2

State Of the Art

2.1 Segmentation

Text Tiling

Automatic textual segmentation is a technique that has been well studied and developed for several years. M.Hearst [Hea94] introduced a statistical method using a bag of words to segment text. In his work, segmentation was based on *geometric* distance between vectorial representations of text. M.Hearst demonstrated that only simple preprocessing was needed to perform well and used only a stop list. This method has been extended in a few ways to perform better or to be more robust.

Extended Geometric Methods

N.Masson [FGM98] proposed to extend the notion of distance computation by using lexical information. A lexical network containing relations between words is used to extend the initial vectorial representation of the document. Hence, when a word A is present, a neighbour word B (not in the document) can be introduced in the bag of words to render the lexical proximity that exists between them. The use of such an external resource while improving the technique's robustness introduces language dependency. K.Richmond [RSA97] develops a language independent computation of distances in text. His idea is to exclude non-content words from the vectorial representation; the use of the content words' distribution pattern in documents provides interesting information to detect these content bearing words.

Lexical Chains

Another totally different approach to segmentation uses lexical chains (see [MT94] and [PBCG$^+$04]) to detect subject boundaries. Chains containing sibling words are constructed while traversing the text. Each time a word is considered, it is added to a chain to keep the best lexical cohesion in each of them. Lexical chains are well suited for this purpose as they keep information on the actual context and help to disambiguate when more than one lexical possibility exists for a word. Once the chains are constructed, it is quite easy to find the segment boundaries. At each position in the text, the number of starting chains and the number of ending chains are computed. These measurements give the plausibility of a segmentation at each position of the text and can be used to select the *right* ones.

All these techniques have proved to be reliable. However, some of them are more or less robust or sometimes require non-obvious implementations.

2.2 Topic Extraction

Keyword extraction is often used in documents indexing, which is a key tool for Internet searches and is therefore well developed. A frequent technique is to use the tf.idf weighting [SB88] of the bag of words to select the content bearing words. This technique has been extended in many ways.

Topic detection and tracking (TDT)

Topic detection is an extended keyword extraction technique mostly used in TDT [ACD$^+$98] where topics are tracked in a news stream, each *new* topic is marked and the following news are attached to the corresponding topic. The language models that have been developed for this purpose (see [Nal03] for example) are not really in the scope of this paper, but demonstrate how semantic and lexical information can be used to improve the processing.

Keyword Extraction

Keyword extraction, as previously discussed, only extracts words present in the document. This is sometimes not the desired behaviour as a word alone, out of its context, does not retain as much information. Therefore, keyword extraction can be disappointing for the user when it only presents a few words, whereas it is still good for indexing as the number of keywords does not have to be as restricted for this purpose. A vector extension technique like the one presented by [FGM98] could provide keywords more descriptive of the whole document even out of their context.

Lexical Chains

Barzilay [BE97] proposes to use lexical chains for summarization. Constructing a set of lexical chains for a document grasps context and lexical cohesion. Barzilay uses these chains to extract important sentences from the text. However, we could think of an algorithm using these chains to extract keywords: for the best chain constructed from the document, one or more words representing the information contained in this chain can be selected.

Semantic Similarity

These techniques use external knowledge sources (i.e. lexical networks or ontologies) to compute the lexical cohesion of words. Computing the cohesion between words is a widely open subject, as there are many ways of doing it. Lexicographic distance is an easy way to compute a measure but only represents word cohesion in terms of number of shared letters – which is not always a criterion of semantic closeness. When semantic databases are used to compute the cohesion, a few *formulas* provide metrics for the semantic distance or similarity of two words. [BH01] gives a good survey on the existing metrics.

A project developed in parallel to this one [vdP04] uses Semantic Similarity for keyword extraction. Its goal is to extract keywords present in the text by using, in addition to the relative frequency ratio (RFR) filtering, the semantic context described by the semantic distance between consecutive words.

Unlike our project, there is no will to extract generic concepts higher in the hierarchy. This one is only used to compute the semantic similarity between words. Then, the similarity value is used to create a hierarchical clustering of all the words in the documents. The best clusters are then selected and a representative keyword in each cluster is selected to annotate the document.

Chapter 3

Semantic Resource

3.1 Description

The EDR Electronic Dictionary [Yok95] is a set of resources developed by the Japan Electronic Dictionary Research Institute, Ltd. and maintained by the Communications Research Laboratory (CRL)[1] that can be used for natural language processing. It is composed by:

- an English and a Japanese word dictionary,
- a Concept dictionary,
- an English ↔ Japanese bilingual dictionary,
- an English and a Japanese co-occurrence dictionary,
- an English and a Japanese corpus.

This electronic resource is consequent; it contains 420'115 dictionary entries for 242'493 unique word forms. Compared to the WordNet[2] ontology that contains 203'145 total word senses, EDR contains 488'732 concepts and 507'665 simple hypernym/hyponym links between them.

For the current project, the English and the Concept dictionary were mostly used. The English dictionary provides information on basic and common words:

- lemma (called *HeadWord*),
- pronunciation,

[1] http://www.crl.go.jp/overview/index.html

[2] WordNet is one of the main ontology freely available and is often used in Natural Language Processing researches. www.cogsci.princeton.edu/w̃n/

- grammatical information such as inflection patterns and part of speech tags,
- semantic information that links to the Concept dictionary.

Its main purpose is to make a correspondence between English words and the conceptual dictionary and to provide basic grammatical information. The Concept dictionary can be considered as an ontology, as it represents the hypernym/hyponym links between common concepts. This dictionary does not make distinctions between English and Japanese concepts, as there is no need to think that there is one. Its content is divided in three sub-dictionaries:

- the *HeadConcept* dictionary which provides information on the human meaning of each concept with a gloss or a representative word,
- the classification dictionary, placing each concept in the conceptual hierarchy,
- the description dictionary, containing additional links between concepts that cannot be described by the super/sub classification.

The EDR dictionary is a complex structure with a considerable amount of information that has not been cited here. Reference to the complete documentation for further information is strongly advised [Jap95]. An example of dictionary entry is presented in table 3.1.

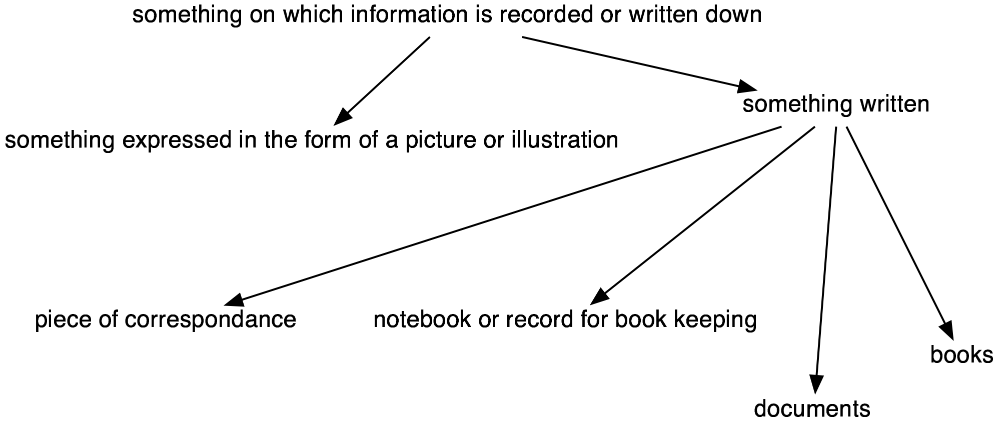
something on which information is recorded or written down
something expressed in the form of a picture or illustration
something written
piece of correspondance
notebook or record for book keeping
books
documents

Figure 3.1: Extract of the concept classification

Even if figure 3.1 presents a simple tree structure extracted from the concept dictionary, it is more frequent to encounter complex structures where hyponyms have more than one hypernym – due to multiple inheritance – and that cannot be

Type	**word**	**syllable**	**pron.**	**POS**	**word form**	**flex.**	**gloss**	F_{word}	$F_{word}^{concept}$
S	Cypriot	Cyp ri ot	s’ipriXet	Com. Noun	Sing.		a Greek dialect used in Cyprus	3	0
S	Cypriot	Cyp ri ot	s’ipriXet	Adj.	Pos. Deg.		pertaining to Cyprus	1	1
S	Cypriote	Cyp ri ote	s’ipriXet	Com. Noun	Sing.	”s” ”-” ”s”	-	0	0
S	Cypriote	Cyp ri ote	s’ipriXet	Adj.	Pos. Deg.		-	0	0
S	Cyprus	Cy prus	s’aiprXes	Prop. Noun	Sing.		an island in the Mediterranean sea named Cyprus	4	2
S	Cyprus	Cy prus	s’aiprXes	Prop. Noun	Sing.		a country called Cyprus	4	2
S	Cyrano de Bergerac	Cy ra no de Ber ge rac	...	Prop. Noun	Sing.		the main character in the play Cyrano de Bergerac	0	0

Table 3.1: Extract of the English Word dictionary

considered as trees. Sometimes, in this document, the conceptual hierarchy will be referred to as the: “Directed Acyclic Graph”.

This data is provided in an structured textual form (see figure 3.2 and 3.3). For ease of use, it has been processed and entered in a database preserving the data structure present in the original format. This processing has been performed in an earlier project at the LIA[3] and facilitates the access to the data. Its major strength is the access API, a feature provided by the querying capacities of any modern database, which makes the development of different applications around EDR faster. A description of the major tables used during the project can be

[3] Artificial Intelligence Laboratory. http://liawww.epfl.ch

found in section 3.3.

3.2 Version Issues

Computerized linguistic resources are rarely as perfect as they should be. Annotating and defining a large quantity of linguistic data requires mostly human work and "error is human". During this project, the use of the EDR database revealed strange values for some of its elements. However it has not been checked whether these errors were in the original data or came from the text to database conversion.

Anonymous Concepts

The documentation of EDR is clear on the fact that, for each *headconcept* present in the hierarchy, there should be at least a gloss – the *headword* is not a must – explaining the concept. Intuitively this seems right: how could the EDR creator have the knowledge to insert a concept (i.e. they know that this concept exists) without being able to define it?

Since a great part of this project relies on extracting concepts to describe a text, having a human description (i.e. a gloss or a *headword*) for each concept is an obligation. However, most of the experiments proved that there is a majority of *anonymous* concepts. Even if they remain important in the construction of the conceptual hierarchy, they must not be displayed to the end user.

Multiple Roots and Orphan Nodes

The EDR documentation is not really clear on what are the ontology's roots. A note in the example about the basic words states that all concepts are located below the concept *#3aa966*. This does not seem to be strictly true. Indeed, all concepts dominated by this root have a **conceptType** of "SIMPLE" but during our experimentation, a second root, the concept *#2f3526*, has been identified. All the concepts dominated by this root have a **conceptType** of "TECHNICAL" which might explains the distinction between the two separate hierarchy.

Counting all the concepts located under these two roots gave a sum of 474'169 concepts, whereas simply counting the number of entries in the *headconcept* databases gave 488'732. 14'563 concepts are neither under the concept *#3aa966* nor under the concept *#2f3526*. These orphan concepts are not linked in the database to any other concepts.

```
<Record Number>                         EWD1364642
<Headword Information>
        <Headword>                      supply
        <Invariable Portion of Headword and Adjacency Attributes Pair>
                                        suppl(Verb with Initial
                                        Consonant Sound, Invariable
                                        Portion of Verb Headword -
                                        Inflection Pattern y)
        <Syllable Division>             sup/ply
        <Pronunciation>                 sXepl'ai
<Grammar Information>
        <Part of Speech>                Verb
        <Syntactic Tree>
        <Word Form and Inflection Information>
        <Word Form Information>         Invariable Portion of Verb
        <Inflection Information>        Inflection Pattern y
        <Grammatical Attributes>
        <Sentence Pattern Information>  Must take a direct object
                                       (direct object is a
                                        phrase);Takes a prepositional
                                        phrase beginning with the
                                        preposition 'to'
        <Function and Position Information>
        <Function Word Information>
<Semantic Information>
        <Concept Identifier>            0ec944
        <Headconcept>
                <Japanese Headconcept>
                <English Headconcept>   supply
        <Concept Explication>
                <Japanese Concept Explication>

                <English Concept Explication>
                                        to supply goods
<Pragmatic and Supplementary Information>
        <Usage>
        <Frequency>                     122/234
<Management Information>
        <Management History Record>     3/4/93
```

Figure 3.2: Example of English Word Dictionary Records (Verb)

```
<Record Number>                         CPH0314159
<Concept Identifier>                    3d0ecb
<Headconcept>
        <English Headconcept>           borrow
        <Japanese Headconcept>          []
<Concept Explication>
        <English Concept Explication>   to use a person's property after
                                        promising to ...
        <Japanese Concept Explication>  ,
<Management Information>
        <Management History Record>     Date of record update "93/04/26"
```

Figure 3.3: Example of Headconcept Records

non-summable frequencies

Each word in the EDR English dictionary has information on its frequency in the EDR corpus. However, a few issues arise; let's take table 3.2 as an example.

invariable portion	**POS**	**concept**	$F_{word}^{concept}$	F_{word}
Alabama	Common Noun	581	2	5
Alabama	Common Noun	582	0	4
Alabama	Common Noun	582	0	5
Alabama	Common Noun	583	0	5
Alabama	Proper Noun	362294	10	10

Table 3.2: The Alabama entries in the EDR database

The word "Alabama" has five entries corresponding to five different concepts. EDR gives a total Frequency for the word (F_{word}) independent from other information (i.e. how many times this word appears in the corpus) and a Frequency ($F_{word}^{concept}$) for the word with one particular concept (i.e. how many times the word was encountered having this concept). Ideally, the $F_{word}^{concept}$ should sum up to F_{word}. However, a few values are wrong (speaking of table 3.2):

- most of the $F_{word}^{concept}$ are null. This means that the word with this concept has never been found in the corpus, but exists anyway in the database. All frequencies should at least be one in order to be used as a realistic value of what is encountered in usual texts.

- F_{word} is not always the same (5, 4 and 10). A word should have a constant frequency that does not depend on other information than the lexicographic form of the lemma.

- the $F_{word}^{concept}$ do not sum to any F_{word}.

3.3 Interesting Tables

The EDR database is divided amongst a few tables, the most important of them representing one of the original *dictionary* structures present in the textual files, the others bearing textual constants. For example, a lemma entry would be searched in the **HeadWords** table, that would return an index to access the English words table (**EWD**).

EWD

The **EWD** table represents the English words dictionary and contains the basic grammatical information and the indexes of the attached concepts. This table is mostly used to generate the lexicon (see section 5.1.1) and to create the initial bag of concepts (see section 5.2). The lemma and the POS are respectively searched in the **HeadWords** and **PartOfSpeechs** tables, the indexes of which are used to select the right row in **EWD**.

CPH

The **CPH** table is one of the concept dictionaries; it basically describes each concept with an *explanation* and a *headconcept.* The *id* field of this table is used in most representations as it makes the link between the **EWD** table, the **CPC** table and this one.

CPC,CPT

The **CPC** table provides the hyponym/hypernym link. Each entry has an *id* – relating to the relation, nothing to do with **CPH** ids– , one *subconcept* id and one *superconcept* id. The entry must be interpreted as "subconcept ← superconcept" and the *id* column ignored.

The **CPT** table represents the other possible links between words and has not been used in this project.

3.4 New Tables

For faster computation in the extraction algorithm (see section 5), offline computations were performed on the existing tables. Results from these computations are stored in two new tables. Here is a quick description of the new data introduced.

the cptscore table

Because we are in a DAG there is more than one path between two concepts. Especially, between a concept and the root, and between a concept and its covered leaves.

Multiple inheritance in an ontology is not really intuitive. We believe that if a same concept have two paths to the root, then the meaning of this concept on a path is not the same as the one on the other path: the entry is unique in the EDR hierarchy but represents two different concepts.

Therefore, the number of leaves covered by a concept must take into account all the meanings of a leaf entry. If a concept has one leaf but three path to this leaf, it is considered to cover three different leaves.

In the same way, we cannot directly compute the path length from one concept to another. Therefore – because we don't know which meaning an entry is considered for – we have to compute an average distance. Moreover, we are interested in the normalized distance between two concepts. In an ontology, there is often some parts that are more informed – with a better granularity in the concept hierarchy – than others because the creators have more knowledge on these parts. A normalized distance between concepts is then more robust to these variations of granularity.

We are interested by the normalized average distance in the DAG between each concept and the root. We can directly compute from the hierarchy (see figure 3.4):

$L_{i,j}$ the distance from the concept c_i to the root following the path j ,

$l_{i,k}$ the distance from a concept c_i to one of its leaves with the path k,

- N the total number of path to the root,
- n the total number of path to the leaves,

We can define the normalized distance to the root as:

$$D(c_i) = \frac{1}{N \times n} \times \sum_{j=0}^{N} \sum_{k=0}^{n} (\frac{L_{ij}}{L_{ij} + l_{ik}})$$

In the same way, we can define the average distance to the covered leaves as:

$$d(c_i) = \frac{1}{N \times n} \times \sum_{j=0}^{N} \sum_{k=0}^{n} (\frac{l_{ij}}{L_{ij} + l_{ik}})$$

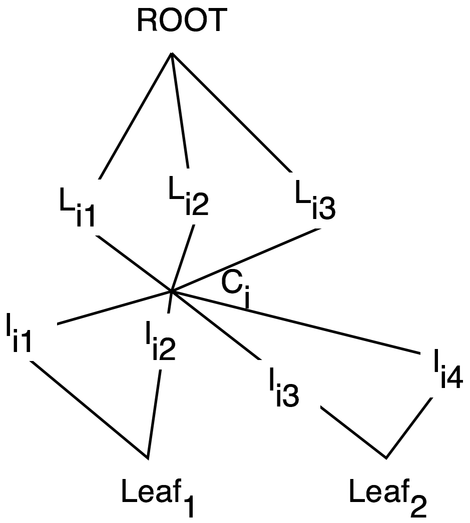

Figure 3.4: Path from a concept c_i

the concept2leaf table

Another important value that is needed for the topic extraction algorithm (see section 5.3.2) is the distance between a concept and each of its leaves. Unlike the distance discussed above, the raw edge count between a concept and each of its covered list must be stored.

Because we can have a lot of path from a concept to all its leaves, this new table is huge. For example, the root has **376'434** entry in this table.

3.5 Corpus and Lexicon

Lexicon Creation

The language processing toolkit developed at LIA [Cha] provides the adequate structures for the creation of a specialized tokenizer and lemmatizer. The first step of this project was therefore dedicated to the production of a lexicon that could be used by this tool; the lexicon had to contain each possible words – in its inflected form – and the corresponding lemmas. The EDR database does not contain the inflected forms of words, but only lemmas with inflection patterns. Therefore, the lexicon was derived from this information.

The EDR dictionary contains 52 possible inflection patterns. Each entry in EDR – the lemmas – is assigned a pattern in function of its POS, however some lemmas have irregular flexions (e.g. "is" cannot be inflected according to regular patterns). In this case, each possible flexion of the lemma is introduced in the database (e.g. “is”, “be”, “was” and “been” will be the three entries for the verb “is”). With 52 inflection patterns applied on the 122’498 regular entries, plus the 132’982 irregular forms already present in the EDR dictionary, 394’703 different flexions are generated for the lexicon. Which is approximately 2 flexions by regular entry. For example “call” has the following inflection patterns:

- ”s-ed” ”-” ”s” ”ed” ”ed” ”ing” as a Verb
- ”s” ”-” ”s” as a Noun

and will be inflected:

- called (Verbs)
- calling (Verbs)
- calls (Nouns)
- calls (Verbs)
- call (Nouns)
- call (Verbs)

Corpus Transformation

The EDR dictionary comes with a corpus of 125’814 short sentences extracted from different journals. These sentences are annotated with different information such as the sentence segmentation, concepts attached to each words or – which is more interesting in this project – the part of speech category of each word.

```
EC00000021  0070000034e6    The Independent 0.8 per cent and now 0.5
per cent.

{   1   0.8 0.8 NUM "=N 0.8"    2   _   _   BLNK    2dc2ed  3
   per_cent    per_cent    NOUN    "=Z percent"    4   _   _   BLNK
   2dc2ed  5   and and CONJ    2dc2f0  6   _   _   BLNK    2dc2ed  7
   now now ADV 0cb281  8   _   _   BLNK    2dc2ed  9   0.5 0.5 NUM "=N
   0.5"    10  _   _   BLNK    ""  11  per_cent    per_cent    NOUN
   "=Z percent"    12  .   .   PUNC    2dc2e5
}

   /1:0.8/2:_/3:per_cent/4:_/5:and/6:_/7:now/8:_/9:0.5/10:_/11:per_cent/12:./
```

```
(S(t(S(S(t(M(S(t(W 1 "0.8"))(W 2 "_"))(t(W 3 "per_cent"))))(W 4
"_"))(t(S(S(t(W 5 "and"))(W 6 "_"))(t(M(S(t(W 7 "now"))(W 8
"_"))(t(M(S(t(W 9 "0.5"))(W 10 "_"))(t(W 11 "per_cent")))))))))(W 12
"."))
```

```
[[main 11:"per_cent":"=Z percent"][time 7:"now":0cb281][number
9:"0.5":"=N 0.5"][and [[main 3:"per_cent":"=Z percent"][number
1:"0.8":"=N 0.8"]]]] DATE="95/7/12"
```

In the perspective of using this annotation as a training base for a Part of Speech tagger [Sch], the corpus was transformed to the correct input format.

However, the set of POS tag in the EDR corpus is quite limited: noun, pronoun, demonstrative, word of negation, question word, intransitive verb, transitive verb, verb, Be-verb, adjective, adverb, adverbial particle, conjunction, prefix, suffix, article, auxiliary, verb, preposition, interjection, blank, space, punctuation symbol, symbol unit, number.

To get a more robust tagger, this tag set must be extended to retain more information on the grammatical context. For example, it is important to know if a verb is at a passive or present form. Therefore, the following set of tags have been chosen:

- an article (ART)
- an auxiliary (AUX)
- a comparative adjective (Adj_Comp)
- a positive adjective (Adj_Pos)
- a superlative adjective (Adj_Super)
- a comparative adverb (Adv_Comp)
- a positive adverb (Adv_Pos)
- a superlative adverb (Adv_Super)
- the verb be (BE)
- a conjunction (CONJ)
- a demonstrative (DEMO)
- Indefinite Pronoun (INDEF)

- interjection (ITJ)
- plural noun (NP)
- singular noun (NS)
- number (NUM)
- prefix (PF)
- proper noun (PN)
- preposition (PREP)
- pronoun (PRON)
- particle (PTCL)
- punctuation (PUNC)
- symbol (SYM)
- to (TO)
- unknown (UKN)
- unit (UNIT)
- basic verb form (VB)
- verb, 3rd person singular (V_3RD)
- verb, past form (V_PF)
- verb, past participle (V_PaP)
- verb, present participle (V_PrP)
- verb, present form (V_Pres)
- wh. pronoun (WH)

Transformation from the original set of tags have been made by following simple heuristics based on the word suffixes – e.g a noun followed by "s" is a plural noun. When these rules cannot resolve the ambiguity on a tag translation – for example, a verb that has the suffix "ed" can either be a past form or a past participle – an existing tagger is used to tag the ambiguous word.

Chapter 4

Segmentation

As seen in the section 2.1, two major segmentation techniques have proved to be efficient. In this project, where the extraction process is bound to linguistic resources, keeping the segmentation *language independent* is not a priority. The lexical chains technique or an extended bag of words representation might therefore be good bases to use the available ontology. However, the aim of this project is more to develop a novel topic extraction technique, thus it was decided to implement a simple Text Tiling method as proposed by M.Hearst [Hea94] to avoid spending time on the issues raised by the use of the EDR ontology (see chapter 3 and section 5.1). The method used is an adaptation of M.Hearst original method to embody the results and recommendations made in [FGM98] (for equations (4.1) and (4.2)) and [RSA97] (for the smoothing, section 4.4).

- The original text is first tokenized and common words (coming from a *stop-list*) are removed. No other preprocessing is performed.

- The text is then divided in preliminary windows, each represented by a weighted bag of words.

- The similarity between each adjacent window is computed and boundaries are placed in accordance to these values.

This process is supported by a linear discourse model inspired by Chafe's notion of Flow Model [Cha79]. This model suggests that the author, when moving from topic to topic, induces a change in the thematic components (i.e. topical words, subject, etc…). The interesting point is that such change is accompanied by a decrease in the lexical cohesion of the text. The Text Tiling algorithm tries to determine this change by using the term repetitions as an indicator of this lexical cohesion. The similarity computation between text windows is then a computation of the lexical cohesion between consecutive blocks of the document. When this similarity reaches a local minimum, a topic shift is detected.

4.1 Preprocessing

Because the segmentation is based on geometric vector distance, it could be interesting to have the highest vector density possible. Indeed, the distance between each vector is a representation of the similitude of their textual content, therefore if the vectors are dense, it will be easier to determine an accurate distance. To obtain denser vectors, it is often better to index them by the word lemma than by the inflected form (i.e. running is similar in meaning to run).

However, to find the correct lemma for a word, it is important to know its part of speech (POS) category and the possible inflections for each category. In the EDR database, the inflection patterns are available and it is easy to derive a lexicon from them; however, at the time the experiments were made on segmentation, no POS tagger for the EDR lexicon was available.

Therefore, having the lemmas without the possibility of disambiguation given by a tagger is not really helpful. In fact, without disambiguation, a considerable irrelevant noise – more than one lemma per word – will be inserted in the vectors, increasing the computation inaccuracy. Moreover, M.Hearst has demonstrated that the use of a simple stop list was enough to remove the common words and to give good results.

4.2 Vectorization And Weighting

After the preprocessing, the document's vocabulary is computed, which contains all the possible words in the document and provides a basis for the vector representation. The document is then segmented into N windows containing the same number of words (a parameter of the algorithm). Each of these windows is represented by a vector $G_i = (g_{i1}, g_{i2}, \ldots, g_{in})$ where g_{ij} is the number of occurrences in the window i of the j^{st} token in the document's vocabulary and n the size of this vocabulary.

Each element of the vector is then weighted using the tf.idf method. The weight w_{ij} is a ratio between the inner-window frequency and the inner-document frequency:

$$w_{ij} = g_{ij} \times \log(\frac{N}{df_j}) \tag{4.1}$$

g_{ij} being the number of occurrences of the word j in the i^{st} window, N the total number of windows and df_j the number of windows where the word j appears.

Hence, the weight of a word spread along the whole document will be lower than the weight of one that occurs in a few windows.

4.3 Similarity Metric

Once the vectors have been weighted, a curve is plotted with the similarity between each adjacent windows along the window number axis.

The similarity is computed using the *Dice Coefficient* defined by:

$$D(G_1, G_2) = \frac{2 \times \sum_{i=1}^{n}(w_{1i} \times w_{2i})}{\sum_{i=1}^{n} w_{1i}^2 \times \sum_{i=1}^{n} w_{2i}^2} \tag{4.2}$$

4.4 Smoothing

A low similarity between two windows can be interpreted as a thematic change whereas a high value represents a continuity in the windows' topic. The figure 4.1 is an example of the similarity variation encountered.

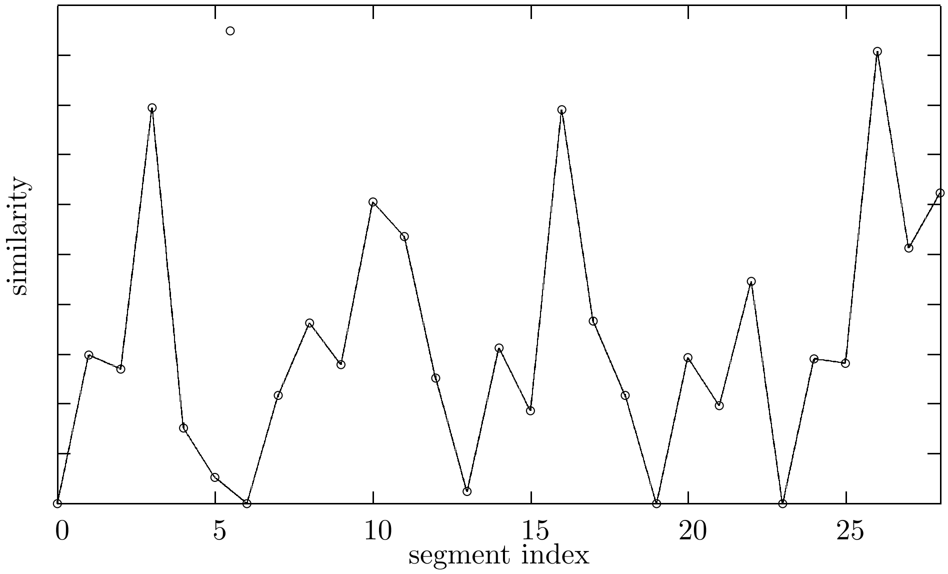

Figure 4.1: Similarity between consecutive windows. (window size: 25, average real segment length: 240 words)

The boundary detection (see next section) is based on the local minima but, as illustrated in figure 4.1, the curve is quite chaotic and contains irrelevant minima. For example, even if the last point has a hight similarity value, it is a local minimum. To prevent the detection of irrelevant boundaries, a simple smoothing is performed on the curve to isolate high value changes from local perturbations.

For each point P_i of the graph, the bisection B of the points P_{i-1} and P_{i+1} is computed and the point P_i is moved in the direction of B by a constant amount. This smoothing is performed on each point in parallel and repeated several times.

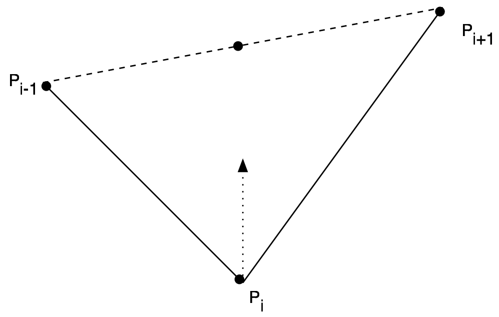

Figure 4.2: Smoothing

Another solution to smooth the similarity curve would have been to overlap each consecutive window so the end of one would have been the start of the next one. Even if this change has not been tested, it is believed to be more robust as the constant shift imposed in the previously described smoothing is another non obvious parameter to the algorithm.

4.5 Boundary Detection

The boundaries for the text can be extracted from the obtained curve by detecting the local minima. To avoid the local perturbations that could have got through the smoothing process, each boundary is given a relevance value. The relevance of the segmentation at the i^{st} point of the curve, is the average of the last local maximum before i and the first local maximum after i. The figure 4.3 is an example (with the same text as in the figure 4.1) of the final result. The boundary detected on a minimum near to its surrounding maxima has a low relevance.

This relevance computation can be used to filter out the extraneous boundaries. This will render the algorithm more robust as a small window size will be applicable to long texts – which contains segments where sub-topic shifts can occur but are not interesting.

4.6 Sentence Detection

Often, the initial windowing, which determines the boundary's positions, is not the best solution. Indeed, window end/start rarely corresponds to what a human

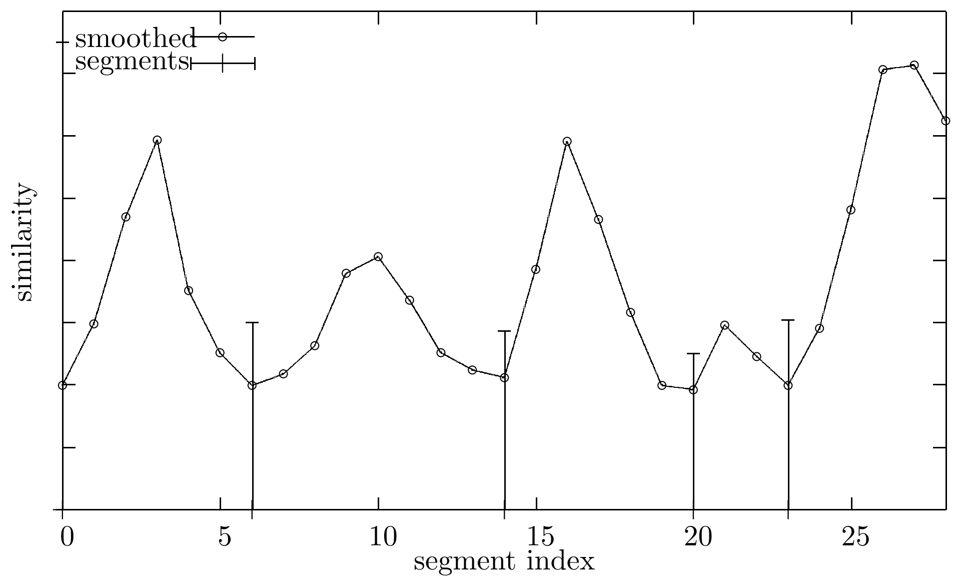

Figure 4.3: A smoothed segmentation. (window size: 25, average real segment length: 240)

would choose as a boundary; punctuation in the text is present as a hint to the reader on low level subject shift, but our algorithm does not have the knowledge to decide if a point is there to end a sentence or to separate a number's digits.

Sentence detection could be implemented using pattern matching; it could then be used to determine the window size – expressed in terms of sentences in place of words – or at the end of the processing to refine the boundary position.

4.7 Sensibility to the Text Type

Two parameters has to be fixed to use this method:

- the window size,
- the smoothing amount.

These two parameters are often difficult to fix automatically as they depend on the text nature. However, they provide a good flexibility to the technique.

The size of the text is highly determinant in the choice of the window size. For long texts, it is better to fix a high window size whereas, for small texts, a lower window size is more applicable. There is more probability than in a small text, the topics will be treated in a smaller number of words. Still, a really low window size can give incorrect results even for small text, see section 6.2 for exegesis of this issue.

This technique, is more relevant to expository texts where the different topics are treated in exclusive blocks of text. Different topics will rarely be mentioned in the same part of the text but will be distributed amongst different sections. For example this report does not mention *segmentation* in the same section as *topic extraction*. Moreover, in scientific or technical texts, it is more probable that the author use the same – technical – term with redundancy.

For narrative texts, the technique is less robust as the author often avoid repetitions. In this kind of texts, it is also natural to mix more than one topic in the same section of the text. Thus, segmentation of this kind of text is less effective and avoiding the detection of too much irrelevant boundaries implies the use of higher window size and of important smoothing, which could drastically decrease the algorithm recall.

Chapter 5

Topic Extraction

Topic extraction can be regarded as an extended keyword extraction task. Keyword extraction is based on statistical methods to identify the content bearing words; this technique is based on Zipf's Law [Zip32].

Relatively frequent words are selected whereas too frequent words (i.e. common words) and low frequency words (i.e. those not really associated to the topic) are discarded as determined by Luhn's upper and lower cut-off [Luh58]. This extraction process has some limits because it only extracts words present in the document. The bag of words extension [FGM98] or the lexical chains techniques [BE97] (see section 2.2) propose some possible solutions. They use an external linguistic resource (i.e. an ontology) to take into account possible hypernyms of the document's keywords.

This project develops a novel method contrasting with the bag of words technique. The novel approach is to give importance to the conceptual position in the ontology during the selection process. When extending the bag of words representation of the document, the existing techniques introduce new words and give them a weight in correspondence to their links with the already present words. Our approach is slightly different, as the initial bag of words representation of the document is not extended but the conceptual hierarchy induced by this representation is constructed and concepts believed to be representative of the document extracted from it.

The aim of this technique is to extract concepts in place of simple words. Concepts provided by the ontology are often associated with a word and/or a gloss (see section 3) and are therefore more interesting. Indeed, the extracted concepts will not necessarily be the leaf ones – i.e. the ones directly triggered by the words present in the document – but more general concepts located higher in the hierarchy. Our selection process tends to favor concepts which are occurs frequently in the document – because some of their sub-concepts are present in the document – and that present an acceptable genericity (see section 5.3.1).

For example, the following text:

> For his fourth birthday, Kevin was brought to the zoo with all his friends. The zoo was invaded by running children. The young child had never seen so many animals which in turn had never seen so many joyful children. During his visit, the child discovered strange cats that were called “lion” or “tiger” and that were quite frightening in spite of their fluffy appearance. The child also discovered the huge mammals: the giant giraffe, the Indian elephant and its African cousin. Kevin and the other children were really impressed by the multicolor parrots and the giant ostrich.
>
> After the walk through the park, all the children were directed to the aquarium where they discovered small fish swimming with threatening sharks. Every child wanted to have a photo with their mechanical replica in the hall.
>
> At the end of the day, they all got an ice cream. Each child was a little crisis of its own when it was his turn to choose the flavor but at the end, all the children were contented and Kevin’s birthday was another dream day in a child life.

will certainly raise the **child** lemma as a representative keyword, because it is well present in the text in the form of “child” and “children”. However, the important topics of **zoo** and **animal** that are mentioned in the text only once or twice will be discarded. However, if the words present were aggregated to more generic concepts, all the mentioned animals will certainly raise the concept **animal** whereas “zoo”, “park” and “aquarium” could raise a concept indicating an **amusement park**.

5.1 Preprocessing

Preprocessing of documents is an issue that has found more than one solution. Two opposed directions are currently chosen in the text mining field:

- either a shallow preprocessing is performed and the consecutive processing will perhaps perform with irrelevant data,
- or the preprocessing is a complex task that drastically decreases the amount of data that can be processed.

The issue with preprocessing is that most of the data generated at the creation of the document will only produce noise in the output of the algorithm. However, there is no efficient way to know what will produce noise and what is interesting in the original data. The first approach avoids loosing too many interesting data in the preprocessing; this is a good way of keeping the output recall high, but lowers the algorithm precision. The second takes the opposite way: the output recall decreases as some interesting data are lost but the precision is higher because the good results are not drowned in the noise.

The first approach has been chosen. A trivial preprocessing (i.e. lemmatization and a stop list) is performed to render the document representation compatible with the EDR database and to remove the common words. Then, using an existing part of speech tagger trained on the EDR data (see section 3.5), part of speech disambiguation is performed.

5.1.1 Tokenization

The initial step in many textual data mining algorithms is the tokenization. This technique is used to split the document in distinct tokens that humans will recognize as words or compounds. In this project, the tokenization had to be performed with the intent of retrieving the entry corresponding to the tokens from the EDR database. More precisely, the available entry points in this dictionary are the word's lemmas and this implies, in addition to the tokenization, the lemmatization of the document. Both steps rely on the use of a lexicon extracted from the EDR dictionary (see section 3.5).

Compound Word Lemmatization

The main problem for standard tokenization is the processing of compounds. Compound words are composed of more than one word and it is difficult to know when it is correct and not to aggregate multiple words to compose one. Complex techniques has been developed to identify compounds in text[CS97]. However, our approach was simpler; because we need entry points in the EDR, the compounds defined in the database were also included during the lexicon creation process. Then, during the tokenization, the tokens are identified as the longest character chain that can be matched with an entry in the lexicon. Hence, whenever it is possible the longest compound is selected.

Making the Lemmatizer More Robust

One obstacle encountered with this technique is that EDR provides some compound proper nouns; these words are either capitalized on the first letter of the first word or on the first letter of all words. While processing the documents, it's frequent to encounter the same compounds with a different capitalization and it's not really efficient to try all possible capitalizations. This issue is associated to the more general question of the presence of proper nouns in the ontology. Most of the proper nouns encountered in a document are not present in EDR and can therefore not be processed by our algorithm; and even if they are present, they are attached to irrelevant concepts. For example, a first name like Michael is attached to the concept: "in the Bible, the archangel who led a group of loyal angels against Satan"; which is rarely what is meant in the document. In some cases, these nouns are the real topics of the text and it is incorect not to take them into account because they are absent from the resource used. Thus implementation of one of the existing proper name identification technique [WRC97] would be optimal.

I saw the man singing with girls in Central Park. The system output is ready.

I(Pron.)— saw(Verb)— the(Det.)— man(Noun)— sing(Verb)— with(?)— girl(Noun)— in(Adv.)— central(Adj.)— park(Noun)— the(Det.)— system output(Noun)— is(Verb)— ready(Adj.)

Figure 5.1: Example of a tokenization/lemmatization

Sadly, there was no simple solution to the capitalization and proper noun identification. Thus, in this project, some interesting words are discarded because the tokenizer is unable to identify a corresponding entry in the database. This is often a cause of loss in precision – when a proper name is not identified correctly – and recall – when a proper name is completely discarded – of the output. In other cases, simple lemmas are identified in place of the real proper names. In the example illustrated in figure 5.1, "system output" which is not capitalized is identified as a compound, whereas "Central Park" – which is a proper name not present in EDR – is only identified as the individual lemmas "central" and "park", which do not retain the proper name meaning.

5.1.2 Part of Speech and Word Sense Disambiguation

Disambiguation is one of the main goals of preprocessing (see [Bri95] and [VI98]): removing some undesired meanings of the lemmas we have found. Indeed, a word alone raises a lot of ambiguity in the algorithm:

- an inflection can be produced from more than one lemma; "left" can be the adjective "left" – as opposed to right – or the simple past of the verb "leave".

- a lemma can correspond to more than one POS category; "run" could be a Noun or a Verb.

- a lemma with a defined part of speech can represent more than one concept; a "controller" could be:
    - a device which controls certain subjects,
    - an executive or official who supervises financial affairs,

There are no perfect solutions to these problems as explained before; too much disambiguation would inevitably reduce the algorithm recall and therefore its accuracy. The first problem is quite unsolvable without any information on the word's context, however, a POS tagger can help in resolving this type of ambiguities. The last ambiguity could be dealt with by using existing technique, however, in this project, no word sense disambiguation was implemented as it was out of it's scope.

Part Of Speech Disambiguation

POS disambiguation corrects the second type of ambiguities – and in some cases the first ones. POS tagging [Bri95] is a state of the art technique that is known to provide very good performances. Using a machine learning approach trained on a large corpus, the tagger can deduce the POS category of each word in the document. In some cases, there is still some ambiguity, but the choices are drastically reduced.

However, the tagger requires a large training corpus where each token is annotated with the correct POS tag. In this project, the corpus used needs to be the same as the one used to construct the EDR database as the tokenization must correspond. An existing tagger [Sch] was trained on a corpus adapted from the EDR data (see section 3.5) but no evaluation was performed on the robustness of this training.

As explained in section 3.5, a new tag set has been chosen. Transformation from this new tag set to the EDR dictionary set to get an entry point in the hierarchy should have been trivial, but in EDR the dictionary tag set is different from the corpus one. Hence, a transformation table was constructed. This transformation table takes a lemma and its POS – which are returned by the tagger – to transform them in an entry in the dictionary.

This is done by mapping the tagger POS to all its possible *inflection pattern/word form/edr pos* triplet in the EDR database. Then, a unique entry that has these grammatical information and the lemma returned by the tagger can be found in the database.

Word Sense Disambiguation

The third case of ambiguity is frequent in our processing, as several concepts are associated to one lemma in the EDR database. This ambiguity is really hard to resolve during the preprocessing, since there is not enough contextual information to discern which sense is the best. The extraction algorithm is in fact the one that should perform the conceptual disambiguation.

This process, known as Word Sense Disambiguation [VI98], is another leading research in the field of Natural Language Processing. Identifying the correct concept attached to a word in the document is a complex task that requires large linguistic resources and time consuming algorithms. For example, one method [DR02] – proposed by M.Diab and P.Resnik – uses unsupervized learning on aligned bilingual corpora to perform disambiguation. Word sense disambiguation is therefore a hard task that requires fully dedicated algorithm and resources. Hence, WSD disambiguation has been left apart for a future project.

5.2 Document Representation

The preprocessing step leaves the data in a state that is not interesting for the subsequent treatments. The preliminary document representation is a list of words and all the possible lemmas (with part of speech information) selected by the tagger. Each word in the post-preprocessing representation holds information on its position in the text, but no information on concepts they are linked to in the EDR hierarchy.

As it is usual in the textual mining field, the first representation computed from the document is a bag of words. Then, using the lemmas attached to each words, a *bag of concepts* is constructed. Each concept is attached to the lemmas that raised it, this one is then attached to each word that raised the lemma. This could help in the future if the relative frequency of each concept should be used in the extraction algorithm.

5.3 Topic Selection

Once the initial conceptual representation of the document have been constructed, the novel algorithm will construct the spanning Directed Acyclic Graph over the *bag of concepts*. Each concept in this hierarchy represents a part of the document, in a *more or less generic* way.

The goal of the algorithm is to extract a set of concepts that will represent the entire document. This set must retain the information contained in the initial set of concepts but it should also provide a more generic representation. For example, all the leaves of the hierarchy are a possible representation, but there is no genericity gained with respect to the initial representation. Another selection in the hierarchy could be its root, this concept is the most generic representation of the document, however, it does not retain anything from the information contained in the document.

These two examples are in fact trivial cuts in a tree: the leaves or the root. A cut in a tree, is a minimal set of nodes that covers all the leaves. The set being minimal, it guarantees that it does not contain a node and one of its subordinates – i.e the cut is not made of nodes that are subordinate with each other.

A cut in our hierarchy would then be a good selection of concepts to represent the content of the document. However, the extracted hierarchy is a DAG and cannot guarantee – because of multiple inheritance – that a concept in a cut will not be a subordinate of another concept in this cut (see algorithm 5.1). As discussed in section 3.4, if a concept has multiple inheritance, it is not considered as having the same meaning.

Then it is acceptable to have a concept C that subordinates a concept A and B and choose A and C in the cut. If the concept C is selected, it will happen because of its inheritance from B that have a different meaning than if it was selected as the subordinate of A.

5.3.1 Cut Extraction

The approach of this algorithm is to extract a cut in the hierarchy that will represent our document. Cuts in the hierarchy are scored and the best one is selected. However, in a regular tree with a branching factor b, the number of cut of depth p is defined by:

$$\begin{aligned} \mathrm{C}(p = 1) &= 1 \\ \mathrm{C}(p > 1) &= \mathrm{C}(p-1)^b + 1 \end{aligned}$$

The number of cut is then exponential in the depth of the tree and evaluating the scores of all cuts in the extracted hierarchy is impossible for a real time algorithm. A dynamic programming algorithm, presented in 5.1 was developed to avoid intractable computations.

One cut is a representation of the leaves of the tree. In this context, the score of a cut χ is computed relatively to the leaves covered. Let $\{f_1, f_2, \ldots, f_M\}$ be the set of leaves in the extracted hierarchy – i.e. the initial representation covered by every cut – and c_i the concept that covers the leaf f_i in the actual cut. A score $\mathrm{U}(c_i)$ is computed for each (c_i, f_i) pair to measure how much the concept c_i is representative of the leaf f_i in the cut χ (see next section).

Algorithm 5.1 Cut Extraction Algorithm

```
initiate the **actual** array with the leaves
while actual is not empty do
  for all concept C in actual do
    add every superconcepts of C to array next
    L ← concept local score
    G ← average of C subconcepts score
    if L ≤ G then
      store G as the score of C
      mark that C should be expended
    else
      store L as the score of C
      mark that C should not be expended
    end if
  end for
  copy next to actual
end while
push the root concept on the stack
while stack is not empty do
  C ← pop(stack)
  if C should be extended then
    push all leaves on stack
  else
    select C
  end if
end while
```

An intuitive score, selected for our algorithm, is the average over all leaves of the score given to each concept in the cut:

$$\mathrm{S}(\chi) = \frac{1}{M} \times \sum_{i=1}^{M} \mathrm{U}(c_i, \chi)$$

5.3.2 Concept Scoring

For each concept, a score measuring how much it represents the leaves it covers in the initial *bag of concepts* must be computed. As explained before, this score should reflect the genericity of a concept compared to the leaves it covers, but should also take into account the amount of information that is kept about these leaves in the selected concept.

Genericity

It is quite intuitive that, in a conceptual hierarchy, a concept is more generic than its subconcepts. The higher we are in the hierarchy, the less specific the concept will be. For example, the concept for "animal" is less specific than the concept representing "dog".

In the same time, even if we are in a DAG, the higher a concept is in the hierarchy, the higher is the number of covered leaves. For example, if "cat", "dog" and "snake" are leaves of the hierarchy, the concept "vertebrate" will cover "cat" and "dog" whereas, it's superconcept "animal" will cover "cat", "dog" and "snake". Following this assertion, a simple score S_1 have been chosen to describe the genericity level of a concept. This score will be proportional to the total number of covered leaves n_i for a concept i:

$$S_1(i, \chi) = \mathrm{f}(n_i)$$

Two constraints are introduced:

- $\mathrm{f}(1) = 0$, a concept covering one leaf is not more generic than that leaf,
- $\mathrm{f}(N) = 1$, the root, covering all leaves, is the most generic concept (N being the total number of leaves in the hierarchy).

With a linear function of the form $\mathrm{f}(x) = a.x + b$, the score can be written:

$$S_1(i, \chi) = \mathrm{f}(n_i) = \frac{n_i - 1}{N - 1}$$

Informativeness

Even if the algorithm should select generic concepts, it cannot always choose the most generic ones, as it will always be the root of the hierarchy. The algorithm, has to take into account the amount of information kept by the selected concepts about the document.

Each concept represents, in a cut, a certain number of leaves of the initial representation. If the concept that is selected to represent a leaf is the leaf itself, then no information is lost. In the same way, if the root is selected to describe a leaf, then all the semantic information retained by this leaf will be lost – i.e. the root represents any leaf in the same way.

An intuitive measure to describe such behavior would be to compute the distance – in number of edges – between the concept i and the leaf it is supposed to represent in the cut χ. A second score S_2 is defined for each concept in the cut. The two constraints exposed above are introduced:

- $S_2(leaf, \chi) = 1$ a leaf represents perfectly itself,
- $S_2(root, \chi) = 0$ the root does not describe anything.

As discussed in section 3.4, a concept covers more than one leaf. Let N be the number of path to the root, n the number of covered leaves (see figure 3.4) in the document hierarchy. The average normalized separation between a concept and its leaves can then be defined as:

$$d(i) = \frac{1}{N \times n} \times \sum_{j=0}^{N} \sum_{k=0}^{n} (\frac{l_{ij}}{L_{ij} + l_{ik}})$$

This measure is equal to 1 for the root and 0 for all leaves. Therefore, S_2 can be computed with:

$$S_2(i, \chi) = 1 - d(i)$$

Score Combination

Two scores are computed for each concept in the evaluated cut, however, a unique score $\mathrm{U}(c_i)$ is needed in the cut scoring scheme. Therefore, a combination formula was chosen for S_1 and S_2. A weighted geometric mean provides a way to render the score independent to the summing formula used in the cut scoring scheme:

$$\mathrm{U}(c_i, \chi) = S_1(i, \chi)^{1-a} \times S_2(i, \chi)^{a}$$

The parameter a is meant to offer a control over the number of concepts returned by the extraction algorithm. If the value of a is near to one, then it will advantage the score S_2 over S_1, and the algorithm will prefer to extract a cut near to the leaves, which will inevitably contain a greater number of concepts. Whereas a value near zero will advantage S_1 over S_2 and therefore prefer more generic concepts in the cut which will be more compact.

Chapter 6

Evaluation

Evaluating the whole system, i.e. performing segmentation and then extraction, would require a testing corpus with complex annotations. A set of texts with segments and their corresponding topics should be used; however, no such corpus is freely available. The evaluation of this project has hence been separated in two different steps; evaluation of the segmentation and evaluation of the topic extraction separately.

6.1 Evaluation Corpora

6.1.1 Reuters Corpus

Segmentation evaluations have been made on the Reuters corpus[1]. The Reuters corpus is made of 806'791 pieces of news, including all English Reuters' news edited between 20/08/1996 and 19/08/1997. The important characteristic of this corpus is that it is freely available to the research community while having a professional content and meta-content.

All news are stored in a separate XML file presenting rich information about the content of the news. Annotation is given about included topics, mentioned countries and industries, for all of the news. These annotations were either automatically or hand checked and corrected to produce an accurate content. Topic codes and industry codes are organized in a hierarchy to produce more complete information.

The annotation is a limited set of topics that features more information on the text category than on its discussed topics. Therefore, it was not used in evaluating the extraction algorithm but, as it provides a quantity of small stories that can be concatenated, it has been used to test the segmentation.

[1] http://about.reuters.com/researchandstandards/corpus/

Here is an example of an entry:

> Europe's leading car-maker Volkswagen AG insists the hotly debated subsidies it received to invest in former communist east Germany were legal, the Financial Times said in its weekend edition.
>
> The newspaper quoted Volkswagen chairman Ferdinand Piech as saying the group was convinced the funds were legal and would stick to its investment plans in the state of Saxony.
>
> Piech said: We received it (the investment) legally and because we did so, we will invest it correctly. We had the choice of any location: without support, it would not have been this one.
>
> A row over the funds erupted last month when Kurt Biedenkopf, state premier of the east German state of Saxony, paid out 142 million marks to Volkswagen, of which Brussels says 91 million marks were unauthorized.
>
> The Saxony payment followed a European ruling in late June clearing only 540 million marks in east German subsidies for the carmaker out of an initial plan for 780 million.
>
> Volkswagen had threatened to pull the plug on its plans to build plants in the Saxony towns of Mosel and Cheimnitz if the funds were not paid in full.
>
> Saxony, which has a jobless rate of more than 15 percent, is determined to retain the car giant's interest in the region.
>
> European Competition Commissioner Karel Van Miert and German Economics Minister Guenter Rexrodt met in Brussels on Friday to resolve the issue but could agree only to continue their efforts to find a solution.
>
> The Commission is due to discuss the case on September 4.
>
> Bonn and the Commission are to continue their talks.
>
> TOPICS: STRATEGY/PLANS and FUNDING/CAPITAL.

6.1.2 INSPEC Bibliographic Database

Description

INSPEC[2] is a database of bibliographic resources about physics, electronics and computing and is composed of:

[2] http://www.iee.org/publish/inspec/

- physics abstracts,
- electrical & electronics abstracts,
- computer & control abstracts,
- business automation.

A set of 238 abstracts were randomly extracted from this database to create a base for the topics extraction evaluation process. In the database, each abstract is annotated with two set that are interesting for the evaluation:

ID Key Phrase Identifiers, containing words assigned by an indexer. These give a description in an open language form of the content of the document.

DE Descriptors, which is a set of keywords describing the content of the document.

The corpus created is then composed of small texts – i.e. the abstracts – annotated by a set of free language words that can be found in the document, but not imperatively in the abstract. Therefore, it provides a good evaluation base for the extraction algorithm as this one is meant to extract topics that are not required to be found in the text. Here is an example of an entry:

AB The entire collection of about 11.5 million MEDLINE abstracts was processed to extract 549 million noun phrases using a shallow syntactic parser. English language strings in the 2002 and 2001 releases of the UMLS Metathesaurus were then matched against these phrases using flexible matching techniques. 34% of the Metathesaurus names occurring in 30% of the concepts were found in the titles and abstracts of articles in the literature. The matching concepts are fairly evenly chemical and non-chemical in nature and span a wide spectrum of semantic types. This paper details the approach taken and the results of the analysis.

DE knowledge-representation-languages; medical-information-systems; text-analysis;

ID UMLS-Metathesaurus; MEDLINE-; shallow-syntactic-parser; flexible-matching-techniques

Transformation

The extraction algorithm gives as its output a list of concepts in the EDR dictionary. Therefore, the list of words describing the document's content – given in an open language form – cannot be used directly for the evaluation. For each word associated to the abstract, a list of concept have been extracted from EDR.

In this transformation process, either there exists at least one concept in EDR that is directly raised by the word or there is no concept in EDR that describes this word. There can be more than one concept associated to a word – due to multiple word senses – but there is no way to know which one is meant in the annotation, therefore, all the possible senses are chosen. If there is no concept associated to a word and its a compound word, then this one is cut in smaller words using the tokenizer developed for EDR (see section 5.1.1) and the concepts attached to all the compound components are kept.

The corpus created in this process provides a set of small text entry annotated with two sets of concepts:

- the ones directly triggered by a word in the corpus annotation,
- the ones that can be deduced from the compound word segmentation.

The second set of concept has been left aside for the current evaluation, as it is possible that it will add too much noise in the evaluation. For example, if the word "flexible matching techniques" is segmented in the concept representing "flexible", "matching" and "techniques", there will be a lot of noise added by the multiple senses of each word separately – the meaning of the compound is not the sum of all the meanings of its composing words.

For some entries, the ratio between the number of words in the initial annotation and the number of found concepts is two small. Therefore, there is no guarantee that the low precision of the extraction algorithm is generated by a bad extraction or the lack of translated annotation (see figure 6.1). Therefore, the corpus has been limited to the entries that had a good ratio – i.e. in out case, over 0.5.

6.2 Segmentation

Segmentation was evaluated with a simple metric on approximately 2'910 texts. These texts were created by concatenating random sized news from the Reuters corpus. A set of either 5 – for 1'608 texts – or 10 – for 1'302 texts – news were selected and composed the *real segments* of the texts.

Evaluating the boundaries found by the automatic segmentation is then a matter of comparing them with the news segments that were used. This is not an obvious task as it is difficult to know the alignment between the *real segments* and the *found* ones. The problem is even more difficult to solve if the number of *found segments* is not the same as the number of *real segments*.

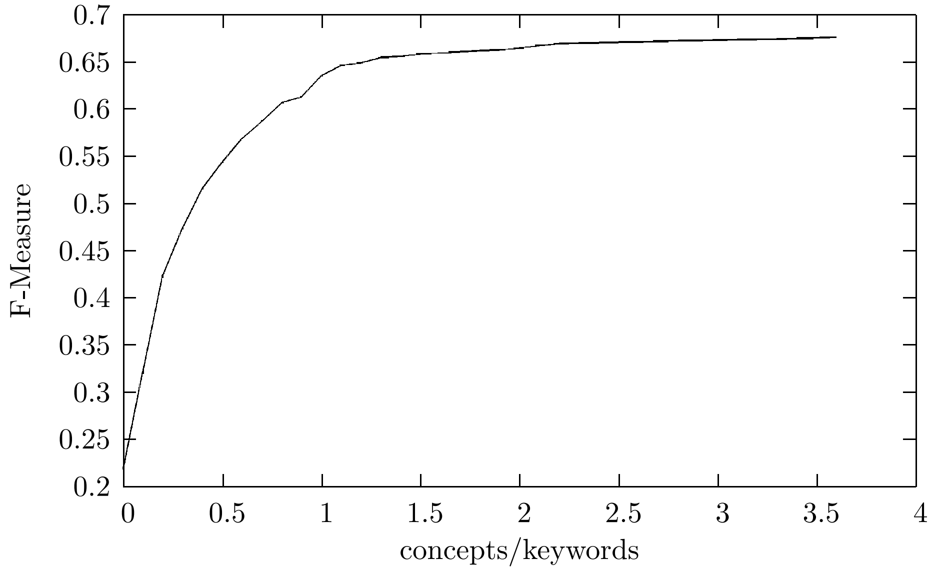

Figure 6.1: F-measure against keywords on found concepts ratio

Metric

A simple metric giving a pseudo error computation has been set up. For each word in the document, its position is computed in both segmentations; for example, in figure 6.2, the positions would be:

$$\underset{\text{real positions}}{D_r = (1,2,2,3,4,4)} \quad \underset{\text{found positions}}{D_f = (1,2,3,3,4,4)}$$

Relative position matrices are computed between each pair of words for both segmentations. The distance $M_{i,j} = |D_j - D_i|$ is computed for each pair of words i and j where word i occurs before word j. Triangular matrices illustrated below are obtained.

$$R = \underset{\text{real distance}}{\begin{pmatrix} 0 \\ 1 & 0 \\ 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 2 & 2 & 1 & 0 \\ 3 & 2 & 2 & 1 & 0 & 0 \end{pmatrix}} \quad F = \underset{\text{found distance}}{\begin{pmatrix} 0 \\ 1 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 1 & 0 \\ 3 & 2 & 1 & 1 & 0 & 0 \end{pmatrix}}$$

The error matrix can then be computed as the difference between the matrices R and F: $E_{i,j} = |R_{i,j} - F_{i,j}|$

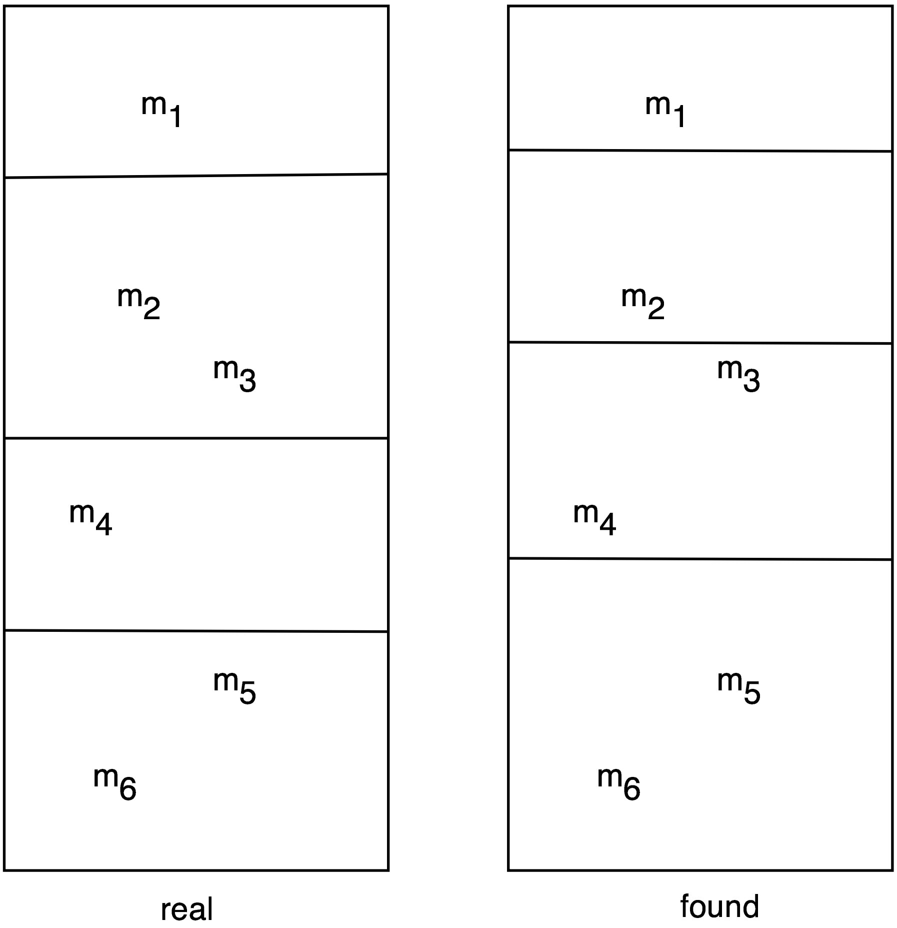

Figure 6.2: Real vs Found Segmentation

$$E = \begin{pmatrix} 0 & & & & & \\ 0 & 0 & & & & \\ 1 & 0 & 0 & & & \\ 0 & 0 & 1 & 0 & & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The average sum of values given by this matrix provides a concrete metric to evaluate the segmentation. This value can be seen as the average error in the placement of a word; if it's high, the segmentation was incorrect, if it is around 0, the segmentation was almost correct.

This value depends of the number of *real* and *found* segments which is linked to the size of the text. If the text is long, there is more chance that there will be more segments (at least *found* ones). If the algorithm finds 20 segments and there are, in reality, only 10 segments, the error value could rise up to 20.

Results

The figures 6.3, 6.4 and 6.5 give plots of the evaluation results. The tests were performed on texts having different size (five and ten segments) and on various window sizes to get a better view of the variations implied by the algorithm parameters.

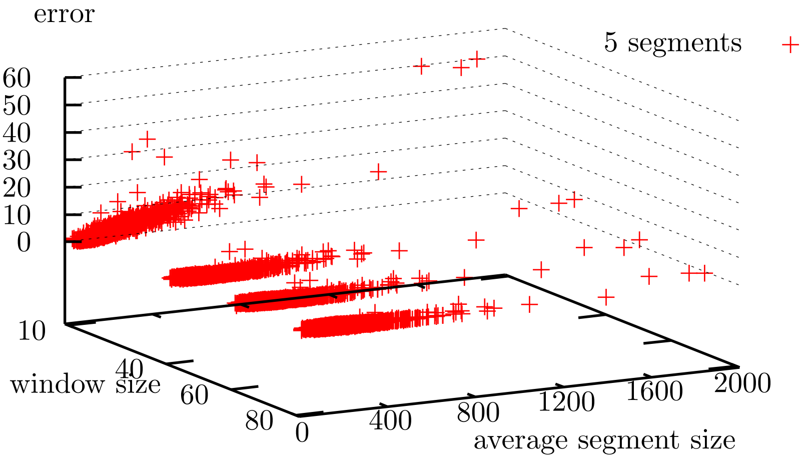

Figure 6.3: Segmentation of 5 segment texts with windows of 10, 40, 60 and 80 words.

Figure 6.3 illustrates the differences made by the window size parameter. This value determines the granularity of the segmentation, for long text, there should be no need to specify a low window size as there will not be many more topics than in small text, but they will be treated in longer segments. For a Window Size of 10, the error rises quickly even for small text as it's certain that the algorithm will detect too many segments (on a small text of 200 words, up to 20 segments could be detected in place of 5).

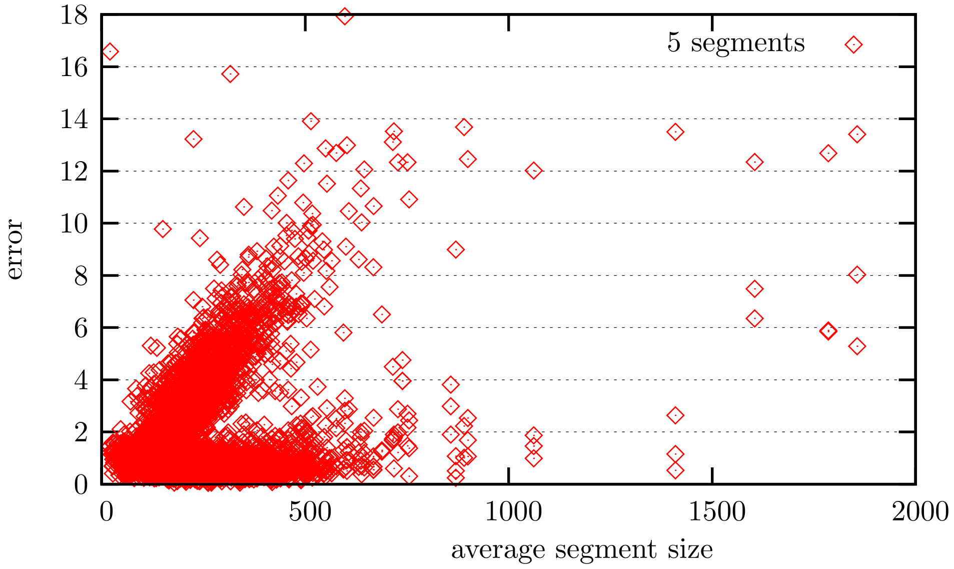

Figure 6.4: Error vs. Segment Size on 5 segment texts with windows of 10, 40, 60 and 80 words.

However, we observe that for windows of size higher than 30, the difference amongst sizes is not so obvious. In figure 6.4, we can identify a group of points corresponding to size 10, but cannot decide in what group of points are the other ones. Small window size generates too many minima that cannot be resolved by the smoothing algorithm (see Section 4.4).

The last figure 6.5 is the most interesting as it clearly shows the robustness of the algorithm with higher window size. It seems that the results for a smaller number of segment are better for windows of size 60 and 80. This could be explained by the fact that on 5 segments there are less chances that two consecutive segments have a similar topic, which will affect the algorithm performance.

interpretation

The obtained results enforce the intuitive thought that a small window size returns bad results. The introduced flow model relies on the fact that the consecutive

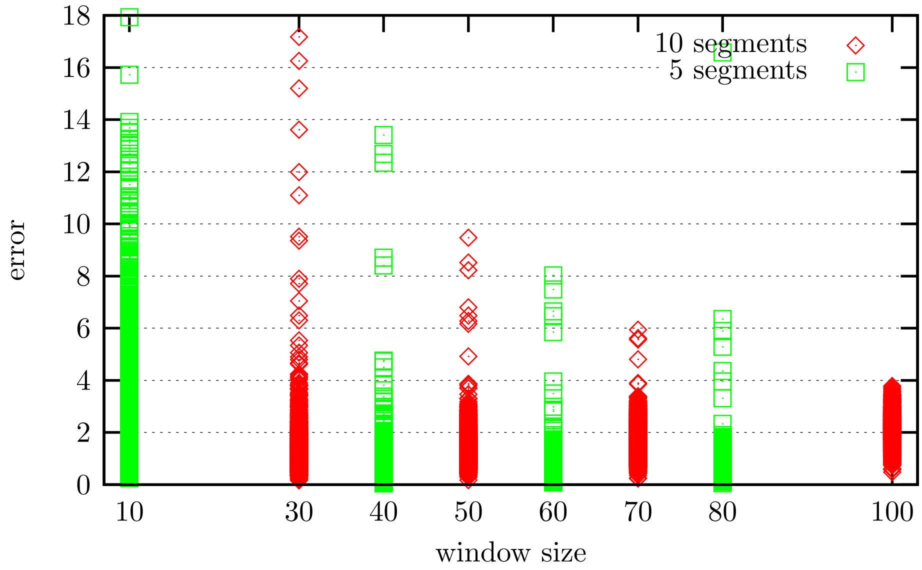

Figure 6.5: Error vs. Window Size for both segmentations.

windows keep a lexical cohesion as long as the topic is not changed. The lexical cohesion computation is made on the criterion of term repetitions; this computation will be wrong for small windows. For example, a window of 10 words barely contains a sentence, and the term repetitions between two consecutive sentences is rarely high as most of the grammatical components change. On larger windows, repetitions are more likely to occur between two consecutive windows and give more flexible variation of the windows' similarity.

Low errors are hard to interpret as it is not obvious how the error is produced, how to know if it corresponds to many misplaced words with a low distance error or a few with a high error. Thus it is not evident to tell what could be done to avoid these small errors. However, there is a high probability – because of the actual implementation – that end or beginning words of certain segments will be badly placed as the window segmentation does not match the sentence segmentation provided by the author.

Segmentation could be slightly improved by refining the window segmentation with criteria like sentence detection. On the other hand, the poorest results mainly come from an inappropriate window size selection which cannot be decided without a basic knowledge of the real segments. This parameter tuning must then be done with the user input. However, choosing a high window size gives a certain robustness and could be sufficient for the approximative segmentation used as preprocessing for topic extraction.

6.3 Topic Extraction

The INSPEC corpus, providing rich information on its documents featured topics, seems to be a choice that would give consistent input on the topic extraction algorithm effectiveness. Moreover, topic annotation in the corpus are generic terms and corresponds perfectly to what is extracted by our algorithm (see the corpus description in section 6.1.2 for more information). Therefore a fully automatic evaluation scheme has been developed over this corpus.

Evaluation Scheme

The evaluation scheme should provide information on the effectiveness of the returned set of concepts compared to the reference annotation in the corpus. But, there is no way to match each concept in the produced set with a concept in the reference set because there is a lot of sibling concepts in our hierarchy that can be interpreted by the user – the annotator in our case – in the same way. For example, if the annotator chose "animal" as an annotation for an entry, but the extraction algorithm returned "vertebrate", there is no big loss in the semantic of the text. However, no automatic algorithm can match these concepts together as it has no knowledge of their meaning.

Table 6.1 presents the standard precision and recall scores obtained with an evaluation that uses the exact match between two concepts –i.e. if the concepts are the same, then a match is found. It's obvious that such evaluation cannot be used in measuring the performances of our topic extraction system.

parameter a	Precision	Recall	F-Measure
0.1	0.0535712	0.0257309	0.030076
0.2	0.0535712	0.0257309	0.030076
0.3	0.0535712	0.0257309	0.030076
0.4	0.0535712	0.0256917	0.0300643
0.5	0.0535712	0.0257033	0.030071
0.6	0.0535712	0.0256987	0.0300641
0.7	0.0540205	0.00890781	0.0145083
0.8	0.061942	0.00823537	0.0136329
0.9	0.0711698	0.01397	0.0209726

Table 6.1: Score with exact matching

Automatic evaluation requires a metric to compare the extracted concepts and the provided topics. Similarity metrics in a concept hierarchy has been used for different purposes in the language processing field and could be performed by a simple edge counting scheme in the EDR hierarchy. [BH01] discusses different metrics that could be used. The Leacock-Chorodow similarity measure [LC98] has been chosen for this evaluation.

This similarity measure is a scaled path length between two concepts. The scaling is done according to the depth of the conceptual hierarchy as a whole. This value is defined by:

$$\mathrm{S}(c_i, C_k) = -\ln\left(\frac{\mathrm{d}(c_i, C_k)}{2D}\right)$$

Where $\mathrm{d}(c_i, C_k)$ is the smallest number of nodes between two concepts and D is the maximal depth of the hierarchy, which is 17 for the EDR dictionary.

The normalized version of this value, noted $\mathrm{p}(c_i|C_k)$ can be interpreted as the probability that the concepts c_i and C_k can be *matched*. The evaluation should give an idea of how much the produced set of concept $\mathrm{Prod} = \{c_1, c_2, \ldots, c_n\}$ is good enough to describe the set of reference $\mathrm{Ref} = \{C_1, C_2, \ldots, C_N\}$.

For each concept c_i the probability that it is *correct* compared to the reference set is the probability that there exists at least one concept in the reference that can be *matched* with c_i. This is the probability that the event "no concept in the reference is *matched* with c_i" does not happen:

$$\mathrm{p}(c_i) = 1 - \prod_{k=1}^{N}(1 - \mathrm{p}(c_i|C_k))$$

In the same way, the probability that at least one concept in *Prod* is matched to the concept C_k is:

$$\mathrm{p}(C_k) = 1 - \prod_{i=1}^{n}(1 - \mathrm{p}(c_i|C_k))$$

For each concept in the produced set, the first probability can be used to compute the quality of the extraction. For example, the probability of *correctness* of an extraction can be computed to determine its accuracy:

$$\mathrm{A}(\mathrm{Prod}, \mathrm{Ref}) = \prod_{k=1}^{n}\mathrm{p}(c_i)$$

However, this value is not always meaningful. For example, if some concepts in Prod are *wrong* – i.e. they do not correspond to any concept in Ref – the score will be low. In the same way, if some concepts in Ref are not represented by a concept in Prod, the score will be low and there will be no way to determine for which of these two reasons the score is *bad*.

In Natural Language Processing, precision and recall are often used to describe the effectiveness of an extraction algorithm. The precision is the ratio between the number of relevant topics extracted and the total number of topics retrieved, whereas the recall is the ratio of relevant topics extracted over the total number of reference topics known.

However, in our evaluation, there is no way to know which concept in Prod is to be matched to a concept in Ref, but we can still describe these score. If we interpret:

$\mathrm{p}(c_i)$ as the probability that the produced concept c_i matches at least one reference concept,

$\mathrm{p}(C_k)$ as the probability that a reference concept C_k is matched by at least one concept produced.

Then the expectation of each score can be computed. Let:

d_i be the function that is equal to 1 when the produced concept c_i is matched and 0 otherwise,

D_k be the function that is equal to 1 when the reference concept C_k is matched and 0 otherwise.

The scores can be written:

$$\begin{aligned}\mathrm{P}(\mathrm{Prod},\mathrm{Ref}) = E(P) &= E\left(\sum_{i=1}^{n}\left(\frac{d_i}{n}\right)\right)\\ &= \sum_{i=1}^{n}\left(\frac{E(d_i)}{n}\right)\\ &= \frac{1}{n}\times\sum_{i=1}^{n}\mathrm{p}(c_i)\end{aligned}$$

and:

$$\begin{aligned}\mathrm{R}(\mathrm{Prod},\mathrm{Ref}) &= E(R)\\ &= E\left(\sum_{k=1}^{N}\left(\frac{D_k}{N}\right)\right)\\ &= \frac{1}{N}\times\sum_{i=k}^{N}\mathrm{p}(C_k)\end{aligned}$$

The usual parametric F-measure can also be derived from these scores:

$$\mathrm{F}(\mathrm{Prod},\mathrm{Ref}) = \frac{(b^2+1).\mathrm{P}(\mathrm{Prod},\mathrm{Ref}).\mathrm{R}(\mathrm{Prod},\mathrm{Ref})}{b^2.\mathrm{P}(\mathrm{Prod},\mathrm{Ref})+\mathrm{R}(\mathrm{Prod},\mathrm{Ref})}$$

Lets look at what is expected for these value. A good algorithm should reach a value of $\mathrm{R}(\mathrm{Prod}, \mathrm{Ref}) = 1$, which means that $\frac{1}{N} \times \sum_{k=1}^{N} \mathrm{p}(C_k)$ is equal to 1, or $\sum_{k=1}^{N} \mathrm{p}(C_k) = N$. Therefore:

$$\begin{aligned} \forall C_k \vdash & \prod_{i=1}^{n} (1 - \mathrm{p}(c_i|C_k)) = 0 \\ \forall C_k \exists c_i \vdash & 1 - \mathrm{p}(c_i|C_k) = 0 \\ \forall C_k \exists c_i \vdash & \mathrm{p}(c_i|C_k) = 1 \end{aligned}$$

That means that for all concepts C_k in the reference set Ref, there exists at least one concept c_i in the produced set Prod that is *equivalent* to C_k. For small values of a, the extracted concepts are generic and there is a small chance that each concept in the reference is perfectly *matched* by a concept in the production.

In the same way, $\mathrm{P}(\mathrm{Prod}, \mathrm{Ref}) = 1$ means that:

$$\forall c_i \exists C_k \vdash \mathrm{p}(c_i|C_k) = 1$$

All the concepts c_i in the production perfectly *match* at least one concept C_k in the reference. For big values of a, the set of concepts extracted contains more specific concepts. It should also contains a bigger number of concepts, therefore, the chances this condition is verified are small –i.e. there is too much noise.

Parameter Optimization

As explained in section 5.3.2, the algorithm is controlled by a parameter a that is supposed to regulate the number of concepts extracted – by controlling the level of the cut in the hierarchy. The corpus that has been created for the evaluation does not always offer the same amount of concepts in each entry annotation. The first experiment to optimize, for each entry, which value of a is used for the evaluation.

In our system, the smallest number of extracted concepts is, the more generic they are. As shown in figure 6.6, nine evaluations have been done on every entry with a different value for the parameter a. This shows the evolution of the algorithm *effectiveness*.

Observing the average results obtained for each value of a, it is obvious that a has a really small impact on the algorithm performances. The F-measure is quasi constant until $a = 0.6$. Then, the Recall rise and the Precision falls because the algorithm outputs a bigger number of concepts that will be more precise – i.e. they match more concepts in Ref but produce more noise.

For value of a under 0.6, the evaluation metric gives comparable results for the similarity of each output with the reference. Which means that the semantic

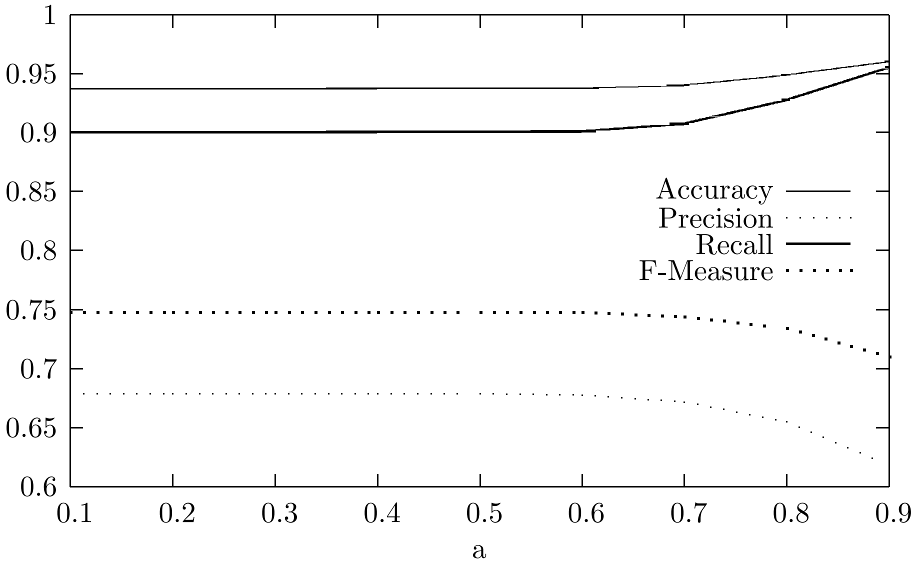

Figure 6.6: Comparison of the algorithm results with variation of a

similarity between each set of extracted concept is small. This could mean two things:

1. the extracted cuts are not very distant in the hierarchy,
2. the similarity metric used considers that the cuts are similar.

Both causes can be explained by looking at the hierarchy extracted during an algorithm run. Such hierarchy is too big to be included in the document, but figure 6.7 displays an example part of one hierarchy. Colored nodes are the ones that are selected in the cut, rectangles are leaves and hexagons are nodes that should be extended (see algorithm 5.1). The little dashed nodes are *anonymous* concepts (see section 3.2) that are ignored in the algorithm processing (an anonymous concepts has no real meaning).

There is many *anonymous* concepts in the hierarchy and if only the non-anonymous ones are considered, the hierarchy is quite flat. The definition of the semantic granularity is poor so the algorithm is unable to select a lot of cut between the leaves and the root. In the same way, there won't be a lot of nodes counted in the similarity metric and many concepts are considered to be *similar*.

With such results, it's hard to choose a best a value for each document to perform the evaluation. Looking at the plot, it would be better to choose a value between $a = 0.1$ and $a = 0.7$ where the results are the bests. However, the plot also displays that the extraction returned *equivalent* results between these values, which means that the extracted concepts are generic ones. We believe that

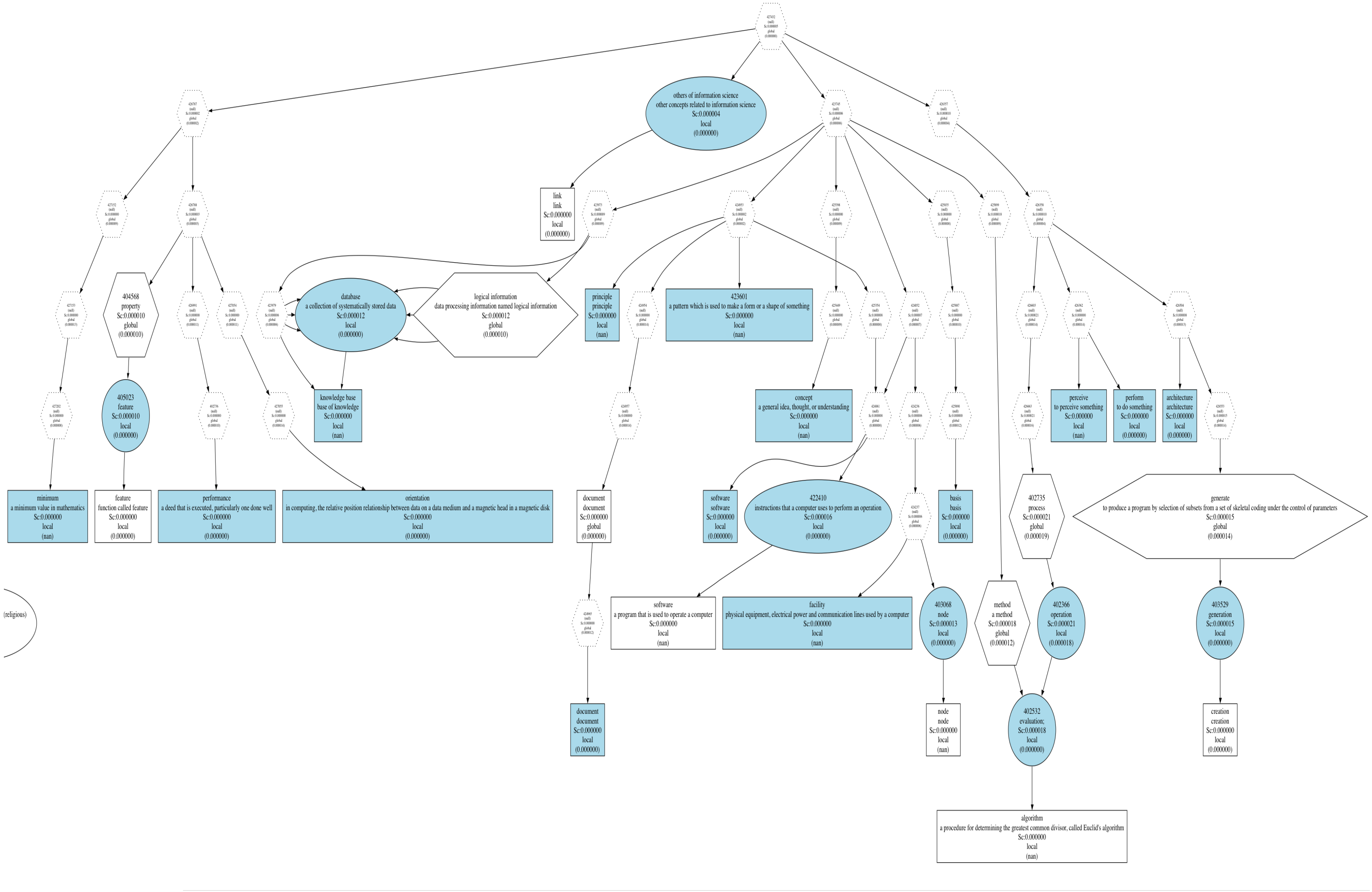

Figure 6.7: Excerpt of an extracted hierarchy

studying the results obtained with a between 0.6 and 0.9 is more interesting as it will display the behaviour of the extraction algorithm when it extract *specific* concepts that might be *nearer* to the reference annotation.

Results Interpretation

A standard evaluation procedure has been adapted to compute the average precision and recall:

1. all the probabilities $p(c_i)$ and $p(C_k)$ is computed for each documents in the evaluation corpus,
2. the concepts c_i in Prod and C_k in Ref is sorted by descending probabilities,
3. for each value of the threshold Θ in [0,1[, a Precision/Recall pair is computed taking into account only the concepts for which $p(c_i) > \Theta$ or $p(C_i) > \Theta$,
4. the average Precision, Recall and F-Measure is computed for each value of Θ.

For example, for a document with the sets of concepts given in table 6.2, and $\Theta = 0.97$, the extracted concepts $\{393453, 358943, 359232, 358845\}$ and the produced concepts $\{22607, 22606\}$ are considered. And the scores are:

$$\mathrm{P}(\mathrm{Pord}, \mathrm{Ref}) = \frac{3.99999629}{4}$$

$$\mathrm{P}(\mathrm{Pord}, \mathrm{Ref}) = \frac{1.96710571}{8}$$

Prod		*Ref*	
c_i	$p(c_i)$	C_k	$p(C_k)$
393453	0.99999954	22607	0.98911961
358943	0.99999950	22606	0.97798610
359232	0.99999866	84062	0.74886709
358845	0.99999859	395795	0.74886709
460912	0.94022170	391576	0.74886709
404568	0.91138175	364931	0.71981943
402831	0.90216440	393495	0.71068739
423743	0.88603944	364267	0.71068739
422997	0.88603944		

Table 6.2: Example of the probability obtained

a	Accuracy	Precision	Recall	F-Measure
0.6	0.93746	0.865545	0.763733	0.78103
0.7	0.940054	0.864756	0.779126	0.79017
0.8	0.948855	0.867199	0.837167	0.825773
0.9	0.960364	0.865917	0.911771	0.870373

Table 6.3: Average results

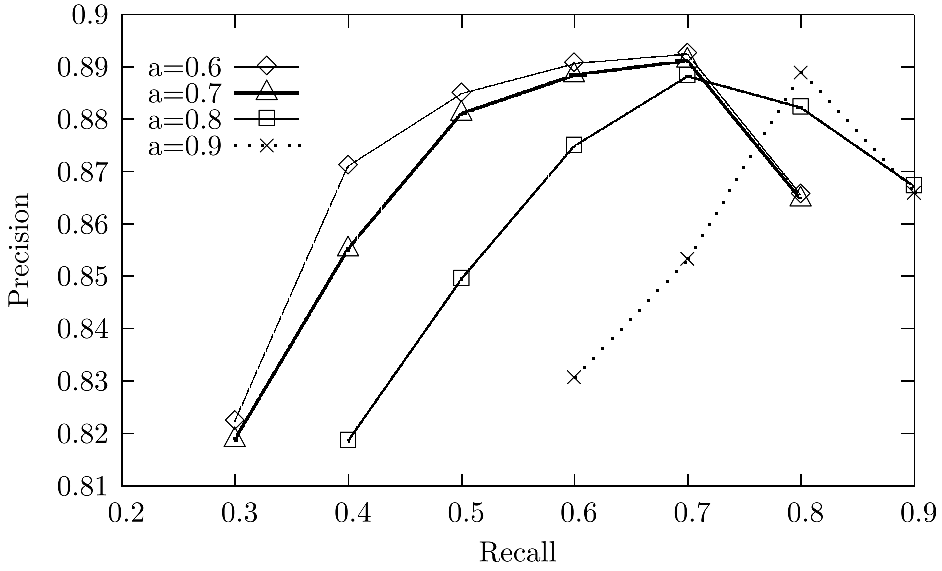

Figure 6.8: Averaged (non-interpolated) Precision/Recall curve

The obtained results are presented in the figure 6.8 and the averages in table 6.3.

With standard keyword extraction algorithm, there is a tradeoff between the precision and the recall. When the recall augments because there is more keywords extracted, the precision will fall because there is more noise in the output. The curve 6.8 displays a – quite interesting – different behaviour. The precision augments in parallel with the recall up to a maximum – around R(Prod, Ref) = 0.7 – where the precision starts to fall.

This is explained by the goal of the algorithm. If we compare a cut χ_1 that contains a small set of concepts and a bigger cut χ_2, there is some chances that the latest contains more concepts because it is lower in the hierarchy than χ_1. That does not always mean that χ_2 is a more *noisy* representation, but it could also be that it is just a more precise set of concepts.

Then, if the recall is raising, it's explained by the fact that the cut is lower in the hierarchy – i.e nearer to the leaves – and has more chances to be near to the reference set of concept. At a point, – around R(Prod, Ref) = 0.7 – the cut is becoming more precise than the reference annotation and the new concepts are considered as *noise*. That's why the precision starts to fall for high values of recall.

A second observation can be made. For $a = 0.6$ and $a = 0.7$, the recall is not defined over 0.9. The discussion at the end of the section 6.3 explains this behaviour. In fact, for small value of a, there is few chances that the extracted cut is specific enough to have a good probability to match all the reference concepts, and therefore it is hard to reach high value of recall.

Chapter 7

Future Work

As mentioned along this report, there is some part that could be continued or added.

7.1 Topic Extraction

Semantic Resource Quality

One main problem with our topic extraction algorithm is presented in the section 3.2 and 6.3. The resource used to construct the semantic hierarchy – which is the base for our computations – and to compute the similarity between concepts – which is the base for our evaluation metric – does not seem to be adapted, by the lack of information stored in the hierarchy, to the task pursued. Adapting the algorithm to use another semantic resource – a newer version of EDR or WordNet for example – may perhaps help in getting a more flexible tool.

In the same way, to render the evaluation independent, the semantic similarity between the produced topics and the reference annotations might be computed from a different semantic resource. This will help getting results not biased by the fact that the hierarchy with which we are evaluating may be erroneous. If the errors lead to a bad extraction, at least, the evaluation will detect that the extraction is erroneous.

Word Sense Disambiguation

In the preprocessing step of the extraction algorithm, there may be many concepts attached to one lemma in the document – due to multiple word senses. This problem could be resolved with two techniques:

- Word Sense Disambiguation, as explained in section 5.1.2.

- In-line disambiguation, that would perform cut-off in the hierarchy to remove parts that are too far from the full hierarchy.

WSD could be performed at preprocessing, by using existing techniques or developing new ones based on the semantic resource used for the extraction. However, too much preprocessing can sometimes lead to loss of precision – i.e if the accuracy of the WSD task is 90% and the accuracy of the extraction task is also 90%, then the system will only perform at 80%, which is not a good result.

In-line disambiguation have been discussed during the project, but the lack of a good resource and of time discouraged us from developing a good implementation. However, there are two main idea that can be used to perform the word sense disambiguation during the extraction process.

The first one will use some semantic similarity metric to remove uninteresting sub-hierarchy from the initial one. If a sub-hierarhcy seems to be too shallow – it does not covers a lot of leaves concepts – or too far from the other side of the full hierarchy – the average semantic similarity between its concept and the other concepts is too low –, it could be removed and a smaller hierarchy will be used to search for cuts.

The second one will use a more complex scoring scheme for the concepts (see section 5.3.2). In the actual scheme, it can be unintuitive to forget about the relative frequency of each concept in the document. It seems interesting to use information about how present a concept is in the document to promote it. Unfortunately, it is not obvious to know how to propagate the frequency of the leaves to their super concepts. This scoring scheme should be explored in more details in future projects.

7.2 Segmentation

The segmentation algorithm used in this project is a really basic implementation of the M.Hearst [Hea94] text tiling technique. Some interesting improvements could be made on it.

The preprocessing task for this algorithm is really basic. Even if M.Hearst has displayed results that show that such preprocessing was sufficient, it could be interesting to study the impact of the addition of a Part Of Speech tagger.

A second step would be the detection and the use in the algorithm of the document structure. For example, the algorithm could create windows (see section 4.2) around full sentences instead of simple words. In the same way, the paragraph segmentation already provided by the document visual formating could be used to put constraints on the window positions.

Our segmentation algorithm makes no use of the semantic resource available.

However, the algorithm could take advantage of the conceptual representation constructed for the extraction algorithm to deduce boundaries. For example, the vector used in the distance computation could be filled with concepts instead of simple lemmas. Then the distance between each conceptual vector will be computed using one of the semantic distance techniques described in [BH01].

Chapter 8

Conclusion

In past systems, Automatic Topic Extraction was mainly treated in two manners where statistical methods and semantic resources were needed. This project combines both techniques to extract interesting topics from a document. A semantic hierarchy is used to extract generic concepts summarizing the text themes. The novel approach introduces the use of simple metrics that stay in the control of the developer as they represent an intuitive manner of choosing representative concepts in a complex ontology.

To reduce the computational bottle neck introduced by the use of a conceptual hierarchy, the text is split in several segments from which topics are extracted. The Text Tiling technique chosen gave good results but revealed a lack of robustness linked to the difficulty of deciding for a general setting of its main parameter: the window size. To obtain the best results, user input is probably needed, but in this project, the segmentation accuracy can stay approximative and one of the high window sizes can be chosen to improve the algorithm robustness.

The segmentation technique could get results improvements by including a sentence detection algorithm to refine the boundaries placement. But it must be kept in mind that this part of the project is to be considered as a preprocessing step. As for most of the preprocessing tasks, it is not always best to spend time refining it when the most interesting part is the topic extraction algorithm.

The topic extraction algorithm is the main novelty in this project and thus offered the possibility to examine interesting concepts and emphasize their strength and weakness. The construction of the spanning Directed Acyclic Graph over the segment's concepts is used to deduce the context of each word and choose the best representative concepts to describe the document's topics.

The novel evaluation method developed to measure extraction algorithm accuracy gives the opportunity to observe interesting behaviours. The evaluation displays good results in term of precision and recall. However, the deficiencies in the semantic resource modified a bit the expected algorithm behaviour. Because the semantic hierarchy is too flat, the user does not have as much control on the extracted concepts genericity as expected when the algorithm was constructed.

The project had the chance to develop interesting and novel parts of a topic annotation system and pointed out important issues and their supposed solutions. Finally two fully usable tools to segment texts and perform topic extraction were implemented in the attempt to contribute with an evolvable code.

This project proves that extracting concepts in place of simple keywords can be beneficial and does not require too complex algorithm. The loss of effectiveness was often coming from erroneous construction of the semantic resource. Hence the development of this technique and of the resources used could bring interesting extensions to existing tools using keyword extraction algorithm.

Document indexing will gain flexibility if the returned documents are conceptually identical to the query and do not simply contain the queried keywords. Multilingual corpora alignment can also gain robustness if they could be aligned on a conceptual basis. However, developing a better semantical topic extraction algorithm will need a flawless linguistic resources and associated tools – i.e. tokenizer, tagger – that are fully compatible with it.

Acknowledgements

I would like to acknowledge here Martin Rajman, the project supervisor, who gave me important directions and interesting inputs. Lonneke van der Plas, Florian Seydoux and Jean-Cedric Chappelier for helping me on different parts of the project.

List of Figures

References

[ACD$^+$98] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report, 1998.

[BE97] R. Barzilay and M. Elhadad. Using lexical chains for text summarization, 1997.

[BH01] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. 2001.

[Bri95] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[Bud99] A. Budanitsky. Lexical semantic relatedness and its application in natural language processing. Technical Report Technical Report CSRG390, University of Toronto., 1999.

[Cha] Jean-Cedric Chappelier. Slptoolkit. http://liawww.epfl.ch/ chaps/SlpTk/.

[Cha79] Wallace Chafe. The flow of thought and the flow of language. In Talmy Givon, editor, *Syntax and Semantics: Discourse and Syntax*, volume 12, pages 159–182. Academic Press, New York, 1979.

[CS97] Jing-Shin Chang and Keh-Yih Su. A multivariate gaussian mixture model for automatic compound word extraction. In *ROCLING-X International Conference*, pages 123–142, Taipei, August 1997.

[DR02] Mona Diab and Philip Resnik. An unsupervised method for word sense tagging using parallel corpora. In *40th anniversary of the Association for Computational Linguistic*, 2002.

[FGM98] Olivier Ferret, Brigitte Grau, and Nicolas Masson. Thematic segmentation of texts: two methods for two kinds of texts, 1998.

[Hea94] Marti Hearst. Multi-paragraph segmentation of expository text. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 9–16, 1994.

[HSO97] G. Hirst and D. St-Onge. Lexical chains as representation of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet: An electronic lexical database and some of its applications.* The mit press. edition, 1997.

[Jap95] Japan Electronic Dictionary Research Institute, Ltd., http://www.iij net.or.jp/edr. *EDR Electronic Dictionary Technical Guide (2nd edition)*, 1995.

[JC97] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, 1997.

[LC98] C. Leacock and M. Chodorow. *WordNet: An electronic lexical database*, chapter Combining local context and WordNet similarity for word sense identification. MIT Press, 1998.

[Lin98] Dekang Lin. An information-theoretic definition of similarity. In *the 15th International Conference on Machine Learning*, Madison, WI, 1998.

[Luh58] H.P. Luhn. The automatic creation of literature abstracts., 1958.

[MT94] Okumura Manabu and Honda Takeo. Word sense disambiguation and text segmentation based on lexical cohesion. In *the Fifteen Conference on Computational Linguistics (COLING-94)*, pages 755–761, 1994.

[Nal03] Ramesh Nallapati. Semantic language models for topic detection and tracking, 2003.

[PBCG$^+$04] A. Popescu-Belis, A. Clark, M. Georgescul, S. Zufferey, and D. Lalanne. Shallow dialogue processing using machine learning algorithms (or not). In H. Bourlard and S. Bengio, editors, *Multimodal Interaction and Related Machine Learning Algorithms.* LNCS, Springer-Verlag, 2004.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[RSA97] Korin Richmond, Andrew Smith, and Einat Amitay. Detecting subject boundaries within text: A language independent statistical approach. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[RSW02] T.G. Rose, M. Stevenson, and M. Whitehead. The reuters corpus volume 1 - from yesterday's news to tomorrow's language resources. In *Third International Conference on Language Resources and Evaluation,*, Las Palmas de Gran Canaria, 2002.

[SB88] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, pages 513–523, 1988.

[Sch] Helmut Schmit. Probabilistic part-of-speech tagging using decision trees.

[SM00] H. Gregory Silber and Kathleen F. McCoy. Efficient text summarization using lexical chains. In *Intelligent User Interfaces*, pages 252–255, 2000.

[vdP04] Lonneke van der Plas. Automatic keyword extraction using the edr and wordnet. Technical report, Interactive Multimodal Information Management, 2004.

[VI98] J. Veronis and N. Ide. Word sense disambiguation: the state of the art. *Computational Linguistics*, 24((1)):1–40, 1998.

[WRC97] N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of proper names in text. In *Fifth Conference on Applied Natural Language Processing*, pages 202–208, 1997.

[Yok95] Toshio Yokoi. The edr electronic dictionary. *Commun. ACM*, 38(11):42–44, 1995.

[Zip32] G.K. Zipf. *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press, Cambridge MA, 1932.

Glossary

A

Application Programming Interface (API) An API is a formally defined programming language interface. It is software that allows application programs to interface to lower level services. p.17

B

bag of words A representation of a text segment. Each word is represented in a vector by a weight, with no information on its placement in the text. Is often used in statistical methods for Textual Data Mining. ... p.8

C

compound A word made up of two or more existing words. Often its meaning is more than the sum of the meanings of the words it is made of. p.35

compounds A word made up of two or more existing words. Often its meaning is more than the sum of the meanings of the words it is made of. p.7

corpus A bank of authentic texts collected in order to find out how language is actually used. Usually a corpus is restricted to a particular type of language use, for example, a corpus of newspaper English, a corpus of legal documents, or a corpus of informal spoken English. p.37

D

Directed Acyclic Graph (DAG) refers to a way of arranging objects based on their relationships and allows a child to have multiple parents. .. p.7

E

eXtensible Markup Language (XML) XML stands for eXtensible Markup Language. It is an extensible document language for specifying document content (i.e. data). XML is not a single, predefined markup language: it's a metalanguage – a language for describing other languages – which lets you design your own markup. p.45

H

hypernym A word that is more generic than a given word. See also: hyponym. p.15

hyponym A word whose meaning denotes a subordinate or subclass, e.g. Dog is a hyponym of animal. Opposite of hypernym. p.15

L

lemma It is the group of all inflected forms of a word. For example, the lemma of all the forms of a verb is the infinitive (e.g., "be" for "is", "were", or "was"). The lemma of a plural is the singular (e.g., "foot" for "feet", or "protein" for "proteins"). p.9

lemmatization A process to extract the set of lemmas that could be derived by grammatical inflections to the set of words processed. p.35

lexical chains A representation of a textual segment where related words are linked together to keep information on relative position and word context. p.11

lexicon Set of words and bound morphemes in a language. A literate speaker understands the phonological, orthographic, and semantic shape (pronunciation, spelling, meaning) of these items and also their morphological and grammatical characteristics. p.23

O

ontology A model of a particular field of knowledge - the concepts and their attributes, as well as the relationships between the concepts. ... p.7

P

part of speech (POS) One of the eight classes of word in English - noun, verb, adjective, adverb, pronoun, preposition, conjunction and interjection. p.16

pattern matching The ability to match a value with any or all characters of an item that is entered as a criterion. A missing character may be represented by a wild card value such as a question mark (?) or an asterisk (*). For example, the request ”find all instances of apple” returns apple, but ”find all instances of apple*” returns apple, applesauce, applecranberry, and so on. p.31

precision Ratio of the number of relevant topics over the total number of topics retrieved.” Precision is a measure of the amount of garbage produced by the system. p.35

R

recall Ratio of relevant topics retrieved for a given document over the number of relevant topics of that document. Recall is a measure of how completely the set of relevant topics was retrieved. p.32

S

semantic The relationships of characters or groups of characters to their meanings, independent of the manner of their interpretation and/or use. p.7

semantic distance A metric to represent the semantic proximity betweens two words or concepts. In most of the cases, it is computed with the help of an ontology. p.13

statistical method A method using only the statistical information available: the inner-document frequency of a word, its frequency in an average text, etc... .... p.8

stop list A list composed of words that are frequent in the common language and rarely bear content in the text. This list contains common words like ”the”, ”go”, etc.... Most of the time, this list is used to reduce the number of words to be taken in account during Textual Mining treatments. p.11

T

tf.idf Textual Frequency times Inverted Document Frequency. p.12

tokenization A process that extract elementary units of a texts: the tokens. A token is a unit of text that cannot be devised in smallest pieces without loosing its meaning in the context of the processing. For example, a token of an English text would be a word or a compound existing in the English vocabulary. p.35

topic detection and tracking (TDT) TDT is the process of segmenting, detecting, and tracking topical information in an information stream. It is essentially a combination of Information Retrieval (detection and tracking) and Speech Recognition (segmentation) technologies. p.12