

Building on Syntactic Annotation: Labelling of Subordinate Clauses

Marina Santini

ITRI

University of Brighton

Marina.Santini@itri.brighton.ac.uk

Abstract:

In this paper, we explore the possibility of labelling semantic/adverbial clauses (i.e. *temporal* clauses, *purpose* clauses, *concession* clauses, etc.), complement/nominal clauses (i.e. *ing*-clauses, *that*-clauses, etc.), and complex noun phrases (NPs) using syntactic patterns built on the syntactic annotation returned by a parser. We suggest that the use of syntactic patterns can represent a useful alternative among other methods for syntax extraction proposed so far. The approach presented here is parser-dependent, but it can be easily adapted to any parser output, for any language. The methodology is simple and it is based on the use of regular expressions. This approach brings about a number of advantages (the use of grammars as a corpus of annotated examples; partial or total disambiguation of ambiguous subordinators; information about the position of the subclause relative to the main clause; labelling of unusual constructions; labelling of several subordinate clauses in the same sentence) which encourage further investigations.

1 Introduction

In this paper, we explore the possibility of labelling semantic/adverbial clauses (i.e. *temporal* clauses, *purpose* clauses, *concession* clauses, etc.), complement/nominal clauses (i.e. *ing*-clauses, *that*-clauses, etc.), and complex noun phrases (NPs) using syntactic patterns built on the syntactic annotation returned by a parser. We consider the first two families of clauses as a kind of subordination, therefore we will call them collectively *subordinate* clauses, including under this cover term also complex NPs which are not, properly speaking, subordinate clauses but complex structures (see Biber et al. 1999: 573 ff.).

The need of labelling subordinate clauses derives from our effort to expand the set of features used in automatic genre and text type identification (see Santini 2004b for a review of the concepts of genre and text types together with the automatic approaches suggested so far). However, this kind of annotation can also be useful in many other NLP tasks, such as the analysis of rhetorical/discoursal strategies, authorship attribution or computational stylistics, natural language generation, information extraction, question answering and so on.

By genres and text types we mean, broadly speaking, a classification of documents which is topic-independent. Genres such as *editorials* or *reviews* can be about any topic, for instance editorials can deal with war, politics, ethics, sports, etc.; reviews can comment on films, books, festivals, etc. The same is true for text types such as *argumentation* or *instruction*, which can be used in any discipline or domain. The idea that different “kinds” of documents entail the use of certain syntactic constructions is not new (Biber 1988: 229-230, Baayen et al. 1996, etc.). However, even if syntax is acknowledged to be revealing (although sometimes reluctantly, see Aaronson 1999), it has often been neglected in genre categorization studies, because the extraction of syntactic features is considered to be computationally expensive and time-consuming (Karlgrén 2000, Kessler et al. 1997). The 67 linguistic features selected by Biber more than 15 years ago (Biber 1988: 73-75, 221-245) are based mainly on the identification of certain lexical items, even when the features are syntactic, because NLP tools were quite limited at that time. For example, he based the identification of adverbial clauses on the presence of specific subordinators, such as *although* for concessive clauses, and *because* for causative clauses. However, the lexically-based approach to syntax is quite limited, because other subordinators can be ambiguous. To overcome ambiguity,

Biber used only unambiguous subordinators; for example *because* is the only causative subordinator included in his features, being the only one "to function unambiguously as a causative adverbial. Other forms, such as *as*, *for* and *since*, can have a range of functions, including causative" (Biber 1988: 236).

More recently, Part-of-Speech (POS) trigrams have been proposed as "shallow" syntactic features, but they are very corpus-dependent (see Argamon et al. 1998 and Santini 2004a for POS trigrams extraction methods), which means that their exportability to other corpora is not guaranteed.

We suggest that the use of syntactic patterns can represent a useful alternative among other methods for syntax extraction proposed so far. In this paper, we concentrate on the automatic labelling of subordinate clauses. Our approach is parser-dependent, but it can be easily adapted to any parser output, for any language (here the working language is English). The methodology is simple and straightforward, as illustrated in Section 3.

This paper is organized as follows: Section 2 briefly reports on neighbouring experiences with syntactic patterns; Section 3 explains the methodology; Section 4 describes two preliminary attempts to evaluate the approach; Section 5 lists the advantages of the approach together with a number of tasks for future work.

2 Previous work

As far as we know, syntactic patterns of any kind have never been tried for genre and text type identification.

Instead, surface or syntactic patterns are regularly used for information extraction (IE) and question answering (QA), as in Ravichandran et al. 2003, Ravichandran and Hovy 2002, Hovy et al. 2002, Soubbotin and Soubbotin 2001, Riloff 1996, etc. The idea behind the use of these patterns is that certain types of answers and certain types of information are expressed using characteristic phrases. Therefore with questions like "When was X born?", typical answers are: "Mozart was born in 1756", "Gandhi (1869-1948)", etc. Patterns for such answers could be: *NAME was born in BIRTHDATE*, *NAME (BIRTHDATE-* (from Ravichandran and Hovy 2002). For extracting information about bombing, a pattern such as *SUBJ was bombed by PP* would return information such as "World Trade Center was bombed by terrorists" (from AutoSlogTS flowchart, Riloff 1996). The purpose of such patterns is to extract snippets of content by making use of some syntactic components to guarantee a certain degree of generalization.

The aim of the syntactic patterns described in this paper, instead, is to label sentences at subclause level and use this kind of annotation for several different NLP tasks.

3 Methodology

3.1 Use of Grammars as a Corpus of Annotated Examples

To our knowledge, there are no public corpora available with annotation at subclause level, such as *manner clause*, *space clause* and so on. For example, the British National Corpus (BNC) includes only morphological annotation (CLAWS-C5 tagset) and the Penn Treebank II Bracketing allows only extraction of predicate-argument structure. All existing public corpora are important and valuable resources, but creating additional manually-checked annotation on the top of existing annotated corpora is very expensive in terms of time and financial support. Of course, it is also a long-lasting resource, as the positive experience of Carlson et al. (2003) shows.

As it is not always possible to fund human annotators, in our study we propose the use of grammars as a corpus of syntactically annotated examples at subclause level. Annotated examples copied from grammars do not require any further annotation or confirmation by humans, as they represent points

of reference of linguistic knowledge. For this task, we used two comprehensive grammars, Quirk et al. (1985), and Biber et al. (1999).

3.2 Steps

The approach to the creation of syntactic patterns for labelling subordinate clauses includes the following steps:

1. Copying examples of subordinate clauses from grammars into file(s). At this stage complement/nominal clause and complex NP were copied from Biber et al. (1999), while semantic/adverbial clauses from Quirk et al. (1985).
2. Parsing the file(s) containing the examples of syntactic constructions. The parser for English used here is Connexor by Tapanainen and Järvinen (1997).
3. Tabulation of the parses in a convenient form, more specifically restoring the horizontal alignment from the Connexor vertical output (see Steps 2 and 3 below).
4. Creation of a set of patterns by identifying the common elements of the parses for each syntactic construction and replacing the optional elements with regular expressions.
5. Creation of an algorithm to identify the sets of patterns in running texts.

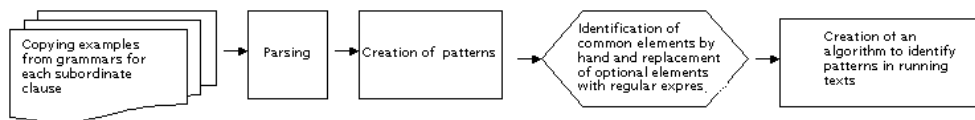


Figure 1. Pipeline for the creation and detection of syntactic patterns.

The five steps will be illustrated using the concessive clause as an example.

Step 1: Grammars as a Corpus of Examples. The Concessive Clause

Examples of clause of concession were copied into a file from Quirk et al. (1985: 1097-1102).

Step 2: Parsing

The file was parsed and a parse was returned for each example.

For instance, the sentence:

```
<!-- : Although he had just joined the company, he was treated exactly like all the other employees.-->
```

was parsed as follows:

#	Text	baseform	Syntactic relation	Syntax and Morphology
1	Although	although	pm:>5	@CS %CS CS
2	he	he	Subj:>3	@SUBJ %NH PRON PERS NOM SG3
3	had	have	v-ch:>5	@+FAUXV %AUX V PAST
4	just	just	meta:>5	@ADVL %EH ADV
5	joined	join	cnd:>11	@-FMAINV %VA EN
6	the	the	det:>7	@DN> %>N DET
7	company	company	obj:>5	@OBJ %NH N NOM
8	,	,		

9	he	he	subj:>10	@SUBJ %NH PRON PERS NOM SG3
10	was	be	v-ch:>11	@+FAUXV %AUX V PAST
11	treated	treat	main:>0	@-FMAINV %VP EN
12	exactly	exactly	man:>11	@ADVL %EH ADV
13	like	like		@ADVL %EH PREP
14	all	all	det:>15	@DN> %>N DET
15	the	the	det:>16	@DN> %>N DET
16	other	other	attr:>17	@DN> %>N DET
17	employees	employee		@NH %NH N NOM PL
18	.	.		
19	<s>	<s>		

Figure 1. A sentence parsed by Connexor.

Step 3: Pattern Creation

Connexor returns several types of annotation¹. For the creation of syntactic patterns, the priority was given to syntax, i.e. the annotation starting with an @ sign, occasionally integrated by other types of annotation, for example the syntactic relation **main:>** that identifies the main verb in the whole sentence, or lexical items, such as **although**, to represent subordinators. Wrong parses were not used to build patterns. A simple algorithm based on string manipulation clears out all unnecessary information and returns the following pattern from the parse shown in Figure 1:

```
<!-- : Although he had just joined the company, he was treated exactly
like all the other employees.-->
```

```
although@CS @SUBJ have@FAUXV @ADVL FMAINV_EN @DN> @OBJ , @SUBJ be@FAUXV
main_FMAINV_EN @ADVL @ADVL @DN> @DN> @DN> @NH
```

Reading such a pattern is very easy:

```
although@CS      = 'although' in the role of subordinate conjunction
@SUBJ           = subject of the clause
have@FAUXV      = 'have' as auxiliary
@ADVL           = adverb
FMAINV_EN       = past participle
@DN>            = determiner
@OBJ            = object
be@FAUXV        = 'be' as auxiliary
main_FMAINV     = main verb in the sentence
@ADVL           = adverb
@NH             = noun head
```

Step 4: Manual Identification of Common Elements across Patterns

What are the common elements between the two following patterns?

¹ The complete annotation scheme is available online at <http://www.connexor.com/demo/doc/enfdg3-tags.html>

Sentence 1:

```
<!-- : Although he had just joined the company, he was treated exactly like all the other employees.-->
```

```
although@CS @SUBJ have@FAUXV @ADVL FMAINV_EN @DN> @OBJ , @SUBJ be@FAUXV main_FMAINV_EN @ADVL @ADVL @DN> @DN> @DN> @NH
```

Sentence 2:

```
<!-- : Although Sam had told the children a bedtime story, June told them one too. -->
```

```
although@CS @SUBJ have@FAUXV FMAINV_EN @DN> @I-OBJ @DN> @A> @OBJ , @SUBJ main_FMAINV_PAST @I-OBJ @OBJ @ADVL
```

It's very easy to detect their similarity: both start with *although*, both have a subject followed by a verb in the subordinate clause, both have a subject and a main verb in the matrix clause. The common elements are:

```
although@CS @SUBJ FMAINV @SUBJ main_FMAINV
```

In order to make this pattern flexible to any number of optional elements occurring between each component, we can simply use regular expressions, provided by many programming languages (the use of regular expression is well-explained in Friedl 1997). The use of a 'non-greedy' quantifier (i.e. a quantifier that finds the minimum number of character matching the pattern) improves efficiency. In many cases, the metacharacters **?* are enough to meet the need of flexibility and efficiency. The pattern filled in by regular expressions has the following form:

```
although@CS.*?@SUBJ.*?FMAINV.*?@SUBJ.*?main_FMAINV
```

and matches both examples shown above.

It is worth highlighting that the added value of a syntactic pattern for an unambiguous subordinator such as *although* is the additional information given by the *position of the subclause relative to the main clause*. In fact, the position of the subordinate clause in the sentence is considered to be genre/register connected and also influenced by coherence and information structuring (cf. Biber et al. 1999: 830-838). The unmarked choice is the final position for all subclause types, therefore a variation of this choice can be informative. For example, semantic/adverbial clauses tend to be in initial position when they contain "given" information referring to previous discourse, while the main clause presents "new" information; on the contrary, when the main clause bears the "given" information, the semantic/adverbial clauses tend to be in final position (see Biber et al. 1999: 835-836).

More importantly, syntactic patterns can help disambiguate ambiguous subordinators, for example *though*. *Though* can be a subordinator and a linking adverbial (Biber et al. 1999:850). With the use of syntactic patterns, the two roles cannot be confused. For example, in the following sentence:

```
No goals were scored, though it was an exciting game.  
@DN> @SUBJ @FAUXV main_FMAINV , though@CS @SUBJ FMAINV @DN> @A> @PCOMPL-S
```

the subclause will be labelled unambiguously as concessive clause, while:

```
He went, though.  
@SUBJ main_FMAINV , though@ADVL
```

will not. Even if the parser makes a tagging mistake, the pattern for *though* subordinator ensures the full disambiguation.

Step 5: Labelling Algorithm

The procedure of finding common elements across patterns was repeated for all examples of subordinate clauses included in this study (see Subsection 3.3). For each subclause, a set of patterns was built. Each set of patterns was searched and labelled by a subroutine, as in the snippet below:

```
sub labelling_concession_clause
{
  undef @initial;
  undef @final;
  undef @special;

  @initial =
  (
    "although\@CS.*?SUBJ.*?FMAINV.*?$s_v_main",
    "_though\@CS.*?SUBJ.*?FMAINV.*?$s_v_main",

    "although\@CS.*?\@NH , .*?$s_v_main",
    "_though\@CS.*?\@NH , .*?$s_v_main",

    "even though\@CS.*?SUBJ.*?FMAINV.*?$s_v_main",
    "even though\@CS.*?NH.*?FMAINV.*?$s_v_main",
    "even though\@CS.*?\@FMAINV_EN.*?$s_v_main",

    "even if\@CS.*?SUBJ.*?FMAINV.*?$s_v_main",
    "even if\@CS.*?NH.*?FMAINV.*?$s_v_main",
  );

  for ($count_features = 0; $count_features<@initial; $count_features++)
  {
    if(/(@initial[$count_features])/)
    {
      print $1;
      print "\t ";
      print "---+> concession_clause_initial\n";
    }
  }
}
```

Figure 2. Example of a subroutine written in Perl to label syntactic patterns of concessive clause.

3.3 Coverage

The subordinate clauses partially or fully covered by syntactic patterns include ten semantic/adverbial clauses, nine complement/nominal clauses, and the complex NP.

The semantic/adverbial clauses are the following:

- | | |
|---|---|
| 1. concession clause (initial, final, special) | 6. reason clause (initial, final) |
| 2. conditional clause (initial, final, special) | 7. result clause |
| 3. contrast clause | 8. similarity manner comparison clause |
| 4. exception clause | 9. space clause (initial, final) |
| 5. purpose clause | 10. time clause (initial, final, incidental, instructional) |

The labels “initial”, “final”, “incidental”, “special” and “instructional” indicate respectively initial position relative to the main clause, final position, incidental position, a special or unusual syntactic construction and, finally, a construction used mainly in instructional texts.

Here is the list of nine complement/nominal clauses, plus the complex NP:

- | | |
|---------------------|--------------------------|
| 1. verb+that clause | 2. adjective+that clause |
|---------------------|--------------------------|

3. that omission
4. wh-clause
5. verb+to clause
6. adjective+to clause

7. verb+ing clause
8. comparative clause
9. relative clause
10. complex NP

The full list of syntactic patterns, with examples, created so far is in Santini (2005:18-37). It is important to point out that these syntactic patterns do not cover all the possible patterns for the subclauses listed above, but only a number of them. Moreover, six subordinators are highly ambiguous (*as, if, since, when, whereas, while*)², therefore for these subordinators only those syntactic patterns which could unambiguously represent a subclause were included. For example, *if* can be both conditional and concessive, but the pattern *if* + past participle, as in the sentence *The grass will grow more quickly if watered regularly* (Santini 2005:27), can never be concessive, so this pattern is unambiguously labelled as conditional. Also the position on the subclause in the sentence can contribute to the disambiguation of subordinators. In the case of *as*, which can mark both temporal and similarity subclauses, the pattern *as* in initial position, followed by a main clause starting with *so*, as in the sentence *As the moth is attracted by a light, so he was attracted by her* (Santini 2005:30), will be unambiguously labelled as a similarity subclause.

Even though these subordinators have not been entirely disambiguated, and the coverage of the syntactic patterns in general is still incomplete, these examples give a flavour of the power and the potential of such an approach. So far, for genre and text type identification, the tendency has been to use only subordinators that are unambiguous, such as *although* for concessive clause or *because* for reason clause (see Introduction), or to overextend the distribution of a subordinator (for example, *if* always conditional, or *when* always temporal). The use of patterns help overcome these limitations.

4 Evaluation

4.1 Initial Assessment of Automatic Labelling

As mentioned earlier, there are no corpora available in English with syntactic annotation at subclause level. Therefore the accuracy of automatic labelling (Step 5 above) cannot be measured against any benchmark. In order to get a rough idea of how well the automatic labelling performs, the syntactic patterns of seven subclauses (*time, manner, purpose, concession, and reason*) created with examples taken from Quirk et al. (1985) were tested against 34 sentences containing the same subclauses taken from Biber et al. (1999).

Figure 4 shows a snippet of the output of the labelling algorithm and is useful to highlight some problems connected to a thorough evaluation of this approach.

² *As* is a subordinator for reason (ex.: *As Jane was the eldest, she looked after the others*), similarity (ex.: *She cooks a turkey as her mother did*) and time (ex.: *As I drove away, I saw her*); *if* for concession (ex.: *If he is poor, he's honest*) and condition (ex.: *If you put the baby down, she will scream*); *since* for reason (ex.: *Since we live near the sea, we often go sailing*) and time (ex.: *Since I last saw her, she has dyed her hair*); *when* for concession (ex.: *She paid, when she could have entered free*), time (ex.: *When I last saw you, you lived in Washington*) and space (ex.: *Take the right fork when the road splits into two*); *whereas* for concession (ex.: *Whereas the amendment was supported in the Senate, its fate is doubtful*) and contrast (ex.: *I ignore them, whereas my husband is worried of what they think of us*); *while* for concession (ex.: *While he has many friends, Peter is lonely*), contrast (ex.: *John teaches physics, while Mary teaches chemistry*) and time (ex.: *He cut himself while shaving*). All the examples come from Quirk et al. (1985).

The output shows different kind of information. For instance, Sentence 1 was copied from Biber et al. 1999 (called LGSWE in the output) on page 818, from examples provided for time adverbials. This information was enriched by the author of this paper with the addition of the position of the subclause relative to the main clause, and the subordinator employed (`<!-- : ***** time_clause_initial_when ***** -->`). Sentence 1 is automatically labelled as *time_clause_initial* and *verb_to_clause*, both labels are correct. One limitation with the use of grammars for evaluation purposes is that it allows the evaluation of only one label per sentence, because the emphasis in grammars is mostly on a single linguistic phenomenon at time. Therefore, the label *verb_to_clause* (*expects to recover*) cannot be objectively evaluated.

As for the other sentences, Sentence 2 receives only the label *complex NP* (*good thoughts*) because the pattern with *until* followed by *be* is not available yet. Sentence 3 is labelled correctly as *time_clause_initial*, and finally Sentence 4 do not receive any label, because, as mentioned before, *as* is a highly ambiguous subordinators, which has not be thoroughly disambiguated yet.

```

<!-- : ===== TIME CLAUSES LGSWE ===== -->
Sentence 1:
<!-- : time_clauses_LGSWE_page_818 - When the units are sold, the city expects
to recover all but its $825,000 initial investment. --><!-- : *****
time_clause_initial_when ***** -->
Pattern 1:
when tmp @ADVL ADV @DN> @SUBJ be@FAUXV V sell tmp @FMAINV EN , @DN> @SUBJ
main FMAINV ---> time_clause_initial
Pattern 2:
factual_verb_main_FMAINV_SG3 @INFMARK> FMAINV ---> verb_to_clause

Sentence 2:
<!-- : time_clauses_LGSWE_page_822 - These good thoughts carry me until I'm
right downtown. --><!-- : ***** time_clause_final_until ***** -->
Pattern 1:
@A> @SUBJ ---> complex_np

Sentence 3:
<!-- : time_clauses_LGSWE_page_822 - When it would not suck from the bottle she
fed it with a teaspoon, fretting with impatience when it coughed and sputtered
and cried. --><!-- : ***** time_clause_initial_when ***** -->
Pattern 1:
when tmp @ADVL ADV it@SUBJ PRON @FAUXV not neg @ADVL NEG suck tmp @FMAINV V
from sou @ADVL PREP @DN> @<P @SUBJ main FMAINV ---> time_clause_initial

Sentence 4:
<!-- : time_clauses_LGSWE_page_822 - Tom could see her in tears as she wrote
it.--><!-- : ***** time_clause_final_as ***** -->

no patterns available yet
[...]

```

Figure 4. A snippet from the output returned by the labelling algorithm.

Three categories were used to give an idea of the quality of the automatic labelling: **C(orrect)**, **I(ncorrect)**, **NAY (not available yet)**. Out of 34 sentences, 21 were labelled correctly, 2 received incorrect labels, and for 11 sentences the patterns were not yet available. If we consider that the patterns for these subclauses were built using only a limited number of examples from Quirk et al. (1985), these results show the good potential of the approach. Presumably, adding new patterns from different grammars or from corpora annotated at subclause level, when they will be available, will extend the coverage of the automatic labelling.

As for the errors performed by the parser, errors were detected especially with special constructions, such as the concessive clause: *Fool that he was he managed to evade his pursuers.*

#	Text	baseform	Syntactic relation	Syntax and Morphology
1	Fool	fool	main:>0	@+FMAINV %VA V IMP
2	that	that	pm:>4	@CS %CS CS
3	he	he	subj:>4	@SUBJ %NH PRON PERS NOM SG3
4	was	be	obj:>1	@+FMAINV %VA V PAST
5	,	,		
6	he	he	subj:>7	@SUBJ %NH PRON PERS NOM SG3
7	managed	manage		@+FMAINV %VA V PAST
8	to	to	pm:>9	@INFMARK> %AUX INFMARK>
9	evade	evade	obj:>7	@-FMAINV %VA V INF
10	his	he	attr:>11	@A> %>N PRON PERS GEN SG3
11	pursuers	pursuer	obj:>9	@OBJ %NH N NOM PL
12	.	.		
13	<p>	<p>		

Figure 5. Wrong parse for a special construction, concessive clause (Quirk et al. 1985:1098).

As you can see in Figure 5, the adjective *fool* is parsed as an imperative. As stated above, patterns are not built from examples parsed incorrectly. The impact of errors performed by the parser will be clear only when a comprehensive benchmark is devised to evaluate the automatic labelling of subordinate clauses.

4.2 Syntactic Patterns as Features for Web Genre Classification

It is important to point out again that syntactic patterns were devised primarily as features for genre and text type identification. To test these features for genre classification, we built two datasets for three web genres: *blogs*, *FAQs*, and *frontpages* (200 documents per web genre). The first dataset was built with normalized frequency counts of syntactic patterns of the following subclasses:

1. adjective_that_clause
2. adjective_to_clause
3. comparative_clause
4. complex_np
5. concession_clause_final
6. concession_clause_initial
7. concession_clause_special
8. conditional_clause_final
9. conditional_clause_initial
10. conditional_clause_special
11. contrast_clause
12. exception_clause
13. purpose_clause
14. reason_clause_final
15. reason_clause_initial
16. relative_clause
17. result_clause
18. similarity_manner_comp_cl.
19. space_clause_final
20. space_clause_initial
21. that_omission
22. time_clause_final
23. time_clause_incidental
24. time_clause_initial
25. time_clause_instructional
26. verb_ing_clause
27. verb_that_clause
28. verb_to_clause
29. wh_clause

The second dataset was build with normalized frequency counts of the following subordinators:

1. although	15. if	29. that
2. as	16. immediately	30. though
3. as_if	17. in_case	31. unless
4. as_long_as	18. in_order_to	32. until
5. as_soon_as	19. just_so	33. what
6. as_though	20. never	34. when
7. because	21. once	35. whenever
8. before	22. only	36. where
9. but_for	23. save_that	37. whereas
10. but_that	24. since	38. wherever
11. even_if	25. so	39. which
12. even_though	26. so_long_as	40. while
13. except	27. so_long_to	41. whoever
14. how	28. so_that	42. why

The learning algorithm used for this task was an SVM classifier as implemented in Weka (Witten and Frank 2000). A 10-fold cross-validation was run 10 times with different seeds, and the accuracy results were averaged. The dataset containing 29 syntactic patterns returned an averaged accuracy of 86.2%, while the dataset with the 42 subordinators reached an averaged accuracy of 83.6%. The gap of 2.6% in the accuracy is quite important if we consider that the list of syntactic patterns is still incomplete, while the list of subordinators is more comprehensive. Presumably, with a better coverage of syntactic patterns the classification accuracy will increase. If so, labelling subordinating clauses appears to be more profitable than using lexical items.

5 Conclusions and Future Work

The approach proposed in this paper brings about a number of advantages, for example:

- 1) the use of grammars as a corpus of annotated examples at subclause level (Subsection 3.1);
- 2) partial disambiguation of some ambiguous subordinators, such as *as* and *if* (Subsection 3.3);
- 3) disambiguation of different syntactic roles (see the example of *though*, Step 4 in Subsection 3.2);
- 4) information about the position of the subclause (initial, final, etc.) relative to the main clause, which can be important for some NLP tasks, such as genre and text type identification, or for a better understanding of information structuring (Step 4 in Subsection 3.2).
- 4) Unusual constructions, like the following concessive clause (Quirk et al. 1985:1098), can be easily identified, without creating any noise:

Naked as I was, I braved the storm.
@PCOMPL-S **as**@CS @SUBJ be_FMAINV , @SUBJ main_FMAINV

In this pattern, there is a subject complement, followed by *as* in the role of conjunction, followed by a subject, followed by a verb *be*, followed by another subject, followed by a main verb. This special construction will be labelled unambiguously as concessive clause.

- 5) Several subordinate clauses can be identified and labelled in a single sentence. For example:

Although Sam had told the children a bedtime story, June told them one too, **because** she was eager to see their reaction.

both patterns:

although@CS @SUBJ.*?FMAINV.*? , @SUBJ.*?main_FMAINV

and

SUBJ.*?FMAINV.*?because@CS.*?SUBJ.*?FMAINV

are applicable, therefore the sentence is fully labelled as bearing a concessive clause in initial position and a reason clause in final position.

All these benefits are encouraging, but lots remain to be done. First of all, a more reliable way to evaluate the automatic labelling must be worked out. As shown in Section 4, each sentence might have several labels. Are human annotators the only solution for this problem? Discussion and suggestions on this specific point are necessary and welcomed. Second, the list of syntactic patterns must be enlarged for the initial set of subclauses used in this study, but also patterns for other subclauses or complex structures must be built. Last but not least, more effort must be put on the full disambiguation of ambiguous subordinators. This is not an easy task, but the linearity of syntactic patterns with the integration of additional linguistic information returned by the parser can help substantially.

6 References

- Aaronson S. (1999), *Stylometric Clustering: A comparative analysis of data-driven and syntactic features*, Project report available at <http://www.cs.berkeley.edu/~aaronson/sc/report.doc>
- Argamon S., Koppel M. and Avneri G. (1998), Routing documents according to style, *Proceedings of the First International Workshop on Innovative Internet Information Systems (IIIS-98)*.
- Baayen H., Halteren (van) H. and Tweedie F. (1996), Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution, *Literary and Linguistic Computing*, 11.
- Biber, D. (1988), *Variations across speech and writing*, Cambridge University Press, Cambridge.
- Biber, D. (1989), A typology of English texts, *Linguistics*, Vol. 27, 3-43.
- Biber D., Johansson S., Leech G., Conrad S. and Finegan E. (1999), *Longman Grammar of Spoken and Written English*, Longman, Harlow.
- Carlson L., Marcu M. and Okurowski M. E. (2003). Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory, in J. van Kuppevelt and R. Smith (eds.) *Current Directions in Discourse and Dialogue*, (Kluwer Academic Publishers), 85-112.
- Friedl J. (1997), *Mastering Regular Expressions*, O' Reilly, Beijing, Cambridge.
- Hovy E., Hermjakob U. and Ravichandran D. (2002), A Question/Answer Typology with Surface Text Patterns, *Proceedings of the DARPA Human Language Technology Conference (HLT)*, San Diego, CA.
- Karlgren J. (2000), *Stylistic Experiments for Information Retrieval*, Thesis submitted for the degree of Doctor of Philosophy, Department of Linguistics, Stockholm University.
- Kessler B., Numberg G. and Shütze H. (1997), Automatic Detection of Text Genre, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*.
- Quirk R., Greenbaum S., Leech G. and Svartvik J. (1985), *A Comprehensive Grammar of the English Language*, Longman.
- Riloff E. (1996), Automatically Generating Extraction Patterns from Untagged Texts, *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*.

- Santini M. (2004a), A Shallow Approach To Syntactic Feature Extraction For Genre Classification, *Proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics* (CLUK 04), University of Birmingham (UK), 6-7 January, 2004.
- Santini M. (2004b), *State-of-the-art on Automatic Genre Identification*, Technical Report ITRI-04-03, 2004, ITRI, University of Brighton (UK).
- Santini M. (2005), *Linguistic Facets for Genre and Text Type Identification: A Description of Linguistically-Motivated Features*, Technical Report ITRI-05-02, 2004, ITRI, University of Brighton (UK).
- Soubbotin M. and Soubbotin S. (2001), Patterns of Potential Answer Expressions as Clues to the Right Answer, *Proceedings of the TREC-10 Conference*, NIST, Gaithersburg, MD.
- Ravichandran D. and Hovy E. (2002), Learning Surface Text Patterns for a Question Answering system, *Proceedings of the 40th ACL conference*, Philadelphia, PA.
- Ravichandran D., Ittycheriah A. and Roukos S. (2003), Automatic Derivation of Surface Text Patterns for a Maximum Entropy Based Question Answering System, *Proceedings of the HLT-NAACL conference*, Edmonton, Canada.
- Tapanainen P., Järvinen T. (1997), A non-projective dependency parser, *Proceedings of the 5th Conference on Applied Natural Language Processing*.
- Witten I. and Frank E. (2000), *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco.