Unsupervised Induction of Sentence Compression Rules

João Cordeiro
CLT and Bioinformatics
University of Beira Interior
Covilhã, Portugal
jpaulo@di.ubi.pt

Gaël Dias
CLT and Bioinformatics
University of Beira Interior
Covilhã, Portugal
ddg@di.ubi.pt

Pavel Brazdil
LIAAD
University of Porto
Porto, Portugal
pbrazdil@liaad.up.pt

Abstract

In this paper, we propose a new unsupervised approach to sentence compression based on shallow linguistic processing. For that purpose, paraphrase extraction and alignment is performed over web news stories extracted automatically from the web on a daily basis to provide structured data examples to the learning process. Compression rules are then learned through the application of Inductive Logic Programming techniques. Qualitative and quantitative evaluations suggests that this is a worth following approach, which might be even improved in the future.

1 Introduction

Sentence compression, simplification or summarization has been an active research subject during this decade. A set of approaches involving machine learning algorithms and statistical models have been experimented and documented in the literature and several of these are described next.

1.1 Related Work

In (Knight & Marcu, 2002) two methods were proposed, one is a probabilistic model - the noisy channel model - where the probabilities for sentence reduction ($P\{S_{compress}|S)\}$ [1]) are estimated from a training set of 1035 $(Sentence, Sentence_{compress})$ pairs, manually crafted, while considering lexical and syntactical features. The other approach learns syntactic tree rewriting rules, defined through four operators: SHIFT, REDUCE DROP and ASSIGN. Sequences of these operators are learned from the training set, and each sequence defines a complete transformation from an original sentence to the compressed version.

In the work of (Le Nguyen & Ho, 2004) two sentence reduction algorithms were also proposed. The first one is based on *template-translation learning*, a method inherited from the *machine translation* field, which learns lexical transformation rules[2], by observing a set of 1500 $(Sentence, Sentence_{reduced})$ pair, selected from a news agency and manually tuned to obtain the training data. Due to complexity difficulties found for the application of this big lexical ruleset, they proposed an improvement where a stochastic *Hidden Markov Model* is trained to help in the decision of which sequence of possible lexical reduction rules should be applied to a specific case.

An unsupervised approach was included in the work of (Turner & Charniak, 2005), where training data are automatically extracted from the Penn Treebank corpus, to fit a noisy channel model, similar to the one used by (Knight & Marcu, 2002). Although it seems an interesting approach to provide new training instances, it still be dependent upon data manually labeled.

More recently, the work of (Clarke & Lapata, 2006) devise a different and quite curious approach, where the sentence compression task is defined as an *optimization* goal, from an *Integer Programming* problem. Several constraints are defined, according to language models, linguistic, and syntactical features. Although this is an unsupervised approach, without using any paralel corpus, it is completely knowledge driven, like a set of crafted rules and heuristics incorporated into a system to solve a certain problem.

1.2 Our Proposal

In this paper, we propose a new approach to this research field, which follows an unsupervised methodology to learn sentence compression rules

[1]In the original paper the $P(t|s)$ notation is used, where t is the sentence in the target language and s the original sentence in the source language.

[2]Those rules are named there as *template-reduction rules*.

based on shallow linguistic processing. We designed a system composed of four main steps working in pipeline, where the first three are responsible for data extraction and preparation and in the last one the induction process takes place. The first step gathers web news stories from related news events collected on a daily basis from which paraphrases are extracted. In the second step, word alignment between two sentences of a paraphrase is processed. In the third step, special regions from these aligned paraphrases, called *bubbles*, are extracted and conveniently preprocessed to feed the induction process. The whole sequence is schematized in figure 1.

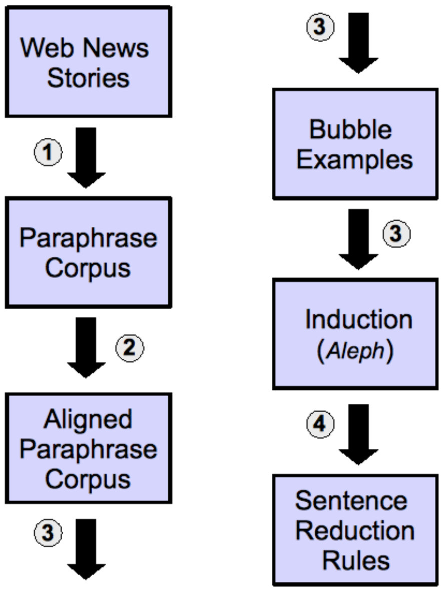

Figure 1: The Pipeline Architecture.

The induction process generates sentence reduction rules which have the following general structure: $L_{cond} \wedge X_{cond} \wedge R_{cond} \Rightarrow suppress(X)$. This means that the sentence segment X will be eliminated if certain conditions hold over left (L), middle (X) and right (R) segments[3]. In Figure 2, we present seven different rules which have been automatically induced from our architecture. These rules are formed by the conjunction of several literals, and they define constraints under which certain sentence subparts may be deleted, therefore compressing or simplifying the sentence. The X symbol stands for the segment

$$Z_{(X)} = 1 \wedge L_c = NP \wedge X_1 = JJ \wedge R_1 = IN \quad (1)$$
$$Z_{(X)} = 1 \wedge L_c = NP \wedge X_1 = RB \wedge R_1 = IN \quad (2)$$

$$Z_{(X)} = 2 \wedge L_1 = and \wedge X_1 = the \wedge R_1 = JJ \quad (3)$$
$$Z_{(X)} = 2 \wedge L_1 = the \wedge X_2 = of \wedge R_1 = NN \quad (4)$$
$$Z_{(X)} = 2 \wedge L_1 = the \wedge X_c = NP \wedge R_1 = NN \quad (5)$$

$$Z_{(X)} = 3 \wedge L_c = PP \wedge X_1 = the \wedge R_c = NP \quad (6)$$
$$Z_{(X)} = 3 \wedge L_c = NP \wedge X_1 = and \wedge R_2 = VB \quad (7)$$

Figure 2: Learned Sentence Compression Rules.

to be dropped, $L_{(\star)}$ and $R_{(\star)}$ are conditions over the left and right contexts respectively. The numeric subscripts indicate the positions[4] where a segment constraint holds and the c subscript stands for a syntactic chunk type. The $Z_{(\bullet)}$ function computes the length of a given segment, by counting the number of words it contains. For instance, the first rule means that a word[5] will be eliminated if we have a NP (Noun Phrase) chunk in the left context, and a preposition or subordinating conjunction, in the right context ($R_1 = IN$). The rule also requires that the elimination word must be an adjective, as we have $X_1 = JJ$.

This rule would be applied to the following segment[6]

```
[NP mutual/jj funds/nns information/nn]
[ADJP available/jj] [PP on/in] [NP
reuters.com/nn]
```

and would delete the word `available` giving rise to the simplified segment:

```
[NP mutual/jj funds/nns information/nn]
[PP on/in] [NP reuters.com/nn].
```

Comparatively to all existing works, we propose in this paper a framework capable to extract compression rules in a real world environment. Moreover, it is fully unsupervised as, at any step of the process, examples do not need to be labeled.

In the remaining of the paper, we will present the overall architecture which achieves precision

[3]For the sake of simplicity and compact representation, we will omit the rule consequent, which is always the same ("$\Rightarrow suppress(X)$"), whenever a rule is presented.

[4]The position starts with 1 and is counted from left to right, on the word segments, except for the left context, where it is counted reversely.

[5]As we have $Z_{(X)} = 1$, the candidate segment size to eliminate is equal to one.

[6]The segment is marked with part-of-speech tags (POS) and chunked with a shallow parser. Both transformations were made with the OpenNLP toolkit.

values up to 85.72%, correctness up to 4.03 in 5 and utility up to 85.72%.

2 Data Preparation

Creating relevant training sets, with some thousands examples is a difficult task, as well as is the migration of such a system to process other languages. Therefore, we propose an unsupervised methodology to automatically create a training set of aligned paraphrases, from electronically available texts on the web. This step is done through step one and step two of Figure 1, and the details are described in the next two subsections.

2.1 Paraphrase Extraction

Our system collects web news stories on a daily basis, and organized them into clusters, which are exclusively related to different and unique events, happening each day: "a company acquisition", "a presidential speech", "a bomb attack", etc. Usually, such clusters contain near 30 small or medium news articles, collected from different media sources. This environment proves to be very fruitful for paraphrase extraction, since we have many sentences conveying similar information yet written in a different form.

A few unsupervised metrics have been applied to automatic paraphrase identification and extraction (Barzilay & Lee, 2003; Dolan et al., 2004). However, these unsupervised methodologies show a major drawback by extracting quasi-exact or even exact match pairs of sentences as they rely on classical string similarity measures such as the *Edit Distance* in the case of (Dolan et al., 2004) and *Word N-gram Overlap* for (Barzilay & Lee, 2003). Such pairs are useless for our purpose, since we aim to identify asymmetrical paraphrase pairs to be used for sentence compression rule induction, as explained in (Cordeiro et al., Oct 2007). There we proposed a new metric, the *Sumo-Metric*, specially designed for asymmetrical entailed pairs identification, and proved better performance over previous established metrics, even in the specific case when tested with the Microsoft Paraphrase Research Corpus (Dolan et al., 2004), which contains mainly symmetrical cases. For a given sentence pair, having each sentence x and y words, and with λ exclusive links between the sentences, the Sumo-Metric is defined in Equation 8 and 9.

$$S(S_a, S_b) = \begin{cases} S(x,y,\lambda) & if \;\; S(x,y,\lambda) < 1.0 \\ 0 & if \;\; \lambda = 0 \\ e^{-k*S(x,y,\lambda)} & otherwise \end{cases} \tag{8}$$

where

$$S(x,y,\lambda) = \alpha \log_2(\frac{x}{\lambda}) + \beta \log_2(\frac{y}{\lambda}) \tag{9}$$

with $\alpha, \beta \in [0,1]$ and $\alpha + \beta = 1$.

We have shown (Cordeiro et al., Oct 2007) that *Sumo-Metric* outperforms all state-of-the-art metrics over all tested corpora and allows to identifying similar sentences with high probability to be paraphrases. In Figure 3, we provide the reader with an example of an extracted paraphrase.

```
(1) To the horror of their fans, Miss Ball
    and Arnaz were divorced in 1960.

(2) Ball and Arnaz divorced in 1960.
```

Figure 3: An Assymetrical Paraphrase

2.2 Paraphrase Alignment

From a corpus of asymmetrical paraphrases, we then use biology-based gene alignment algorithms to align the words contained in each of the two sentences within each paraphrase. For that purpose, we implemented two well established algorithms, one identifying local alignments (Smith & Waterman, 1981) and the other one computing global alignments (Needleman & Wunsch, 1970). We also proposed a convenient dynamic strategy (Cordeiro et al., 2007), which chooses the best alignment algorithm to be applied to a specific case at runtime.

The difference between local and global sequence alignments is illustrated below, where we use letters, instead of words, to better fit our paper space constraints. Suppose that we have the following two sequences: `[D,H,M,S,T,P,R,Q,I,S]` and `[T,P,Q,I,S,D,H,S]` a global alignment would produce the following pair.

```
D H M S T P R Q I S _ _ _
_ _ _ _ T P _ Q I S D H S
```

For the same two sequences, a local alignment strategy could generate two or more aligned subsequences as follows.

```
|D H M S|        |T P R Q I S|
|D H _ S|        |T P _ Q I S|
```

Hence, at this stage of the process, we end with a corpus of aligned[7] asymmetrical paraphrases. In Figure 4, we present the alignment of the paraphrase of Figure 3.

```
(1) To the horror of their fans ,
(2) __ ___ ______ __ _____ ____ _

(1) Miss Ball and Arnaz were divorced in 1960.
(2) ____ Ball and Arnaz ____ divorced in 1960.
```

Figure 4: An Aligned Paraphrase

The next section describes how we use this structured data to extract instances which are going to feed a learning system.

3 Bubble Extraction

In order to learn rewriting rules, we have focus our experiences on a special kind of data, selected from the corpus of aligned sentences, and we named this data as *Bubbles*[8]. Given two word aligned sentences, a bubble is a non-empty segment aligned with an empty segment of the other sentence of the paraphrase, sharing a "strong" context. In Figure 5, we show different examples of bubbles.

```
the situation here in chicago with the workers
the situation ____ in chicago with the workers

obama talks exclusively with tom brokaw on meet
obama talks ___________ with tom brokaw on meet

Ball and Arnaz were divorced in 1960
Ball and Arnaz ____ divorced in 1960

america is in the exact same seat as sweigert and
america is in ___ _____ same seat as sweigert and

after a while at the regents park gym, the president
after a while at ___ _______ ____ gym, the president
```

Figure 5: Examples of Bubbles

To extract a bubble, left and right contexts of equally aligned words must occur, and the probability of such extraction depends on the contexts size as well as the size of the region aligned with the empty space. The main idea is to eliminate cases where the bubble middle sequence is too large when compared to the size of left and right contexts. More precisely, we use the condition in Equation 10 to decide whether a bubble should be extracted or not.

$$Z_{(L)} - Z_{(X)} + Z_{(R)} \geq 0 \qquad (10)$$

where L and R stand for the left and right contexts, respectively, and X is the middle region. The $Z_{(\bullet)}$ function computes the length of a given segment, in terms of number of words. For example, in the first and last examples of Figure 5, we have: $2 - 1 + 5 = 6 \geq 0$ and $4 - 3 + 4 = 5 \geq 0$. In this case, both bubbles will be extracted. This condition is defined to prevent from extracting eccentric cases, as the ones shown in the examples shown in Figure 6, where the conditions respectively fail: $0 - 8 + 3 = -5 < 0$ and $1 - 7 + 2 = -4 < 0$.

```
To the horror of their fans , Miss Ball and Arnaz
__ ___ ______ __ _____ ____ _ ____ Ball and Arnaz

will vote __ ___ _______ ____ __ _____ __ friday .
____ vote on the amended bill as early as friday .
```

Figure 6: Examples of Rejected Bubbles

Indeed, we favor examples with high common contexts and few deleted words to enhance the induction process.

So far, we only consider bubbles where the middle region is aligned with a void segment ($X \overset{transf}{\longrightarrow} \emptyset$). However, more general transformations will be investigated in the future. Indeed, any transformation $X \overset{transf}{\longrightarrow} Y$, where $Y \neq \emptyset$, having $Z_{(X)} > Z_{(Y)}$, may be a relevant compression example.

Following this methodology, we obtain a huge set of examples, where relevant sentence transformations occur. To have an idea about the amount of data we are working with, from a set of 30 days web news stories (133.5 MB of raw text), we identified and extracted 596678 aligned paraphrases, from which 143761 bubbles were obtained.

In the next section, we show how we explore Inductive Logic Programming (ILP) techniques to generalize regularities and find conditions to compress sentence segments.

4 The Induction of Compression Rules

Many different algorithms exist to induce knowledge from data. In this paper, we use Inductive Logic Programming (ILP) (Muggleton, 1991) and it was a choice based on a set of relevant features like: the capacity to generate symbolic and

[7]By "aligned" we mean, from now on, word alignment between paraphrase sentence pairs.

[8]There are other possible regions to explore, but due to the complexity of this task, we decided to initially work only with bubbles

relational knowledge; the possibility to securely avoid negative instances; the ability to mix different types of attribute and to have more control over the theory search process.

Unlike (Clarke & Lapata, 2006), we aim at inducing human understandable knowledge, also known as symbolic knowledge. For that purpose, ILP satisfies perfectly this goal by producing clauses based on *first order logic*. Moreover, most of the learning algorithms require a complete definition and characterization of the feature set, prior to the learning process, where any attribute must be specified. This is a conceptual bottleneck to many learning problems such as ours, since we need to combine different types of attributes i.e. lexical, morpho-syntactic and syntactical. With ILP, we only need to define a set of possible features and the induction process will search throughout this set.

4.1 The Aleph System

The *Aleph* system(Srinivasan, 2000) is an empirical ILP system, initially designed to be a prototype for exploring ILP ideas. It has become a quite mature ILP implementation, used in many research projects, ranging form Biology to NLP. In fact, Aleph is the successor of several and "more primitive" ILP systems, like: Progol (Muggleton, 1999), FOIL (Quinlan, 1990), and Indlog (Camacho, 1994), among others, and may be appropriately parametrized to emulate any of those older systems.

One interesting advantage in *Aleph* is the possibility to learn exclusively from positive instances, contrarily to what is required by most learning systems. Moreover, there is theoretical research work (Muggleton, 1996) demonstrating that the increase in the learning error tend to be negligible with the absence of negative examples, as the number of learning instances increases. This is a relevant issue, for many learning domains, and specially ours, where negative examples are not available.

4.2 Learning Instances

In our problem, we define predicates that characterize possible features to be considered during the induction process. Regarding the structure of our learning instances (bubbles), we define predicates which restrict left and right context sequences as well as the aligned middle sequence. In particular, we limit the size of our context sequences to a maximum of three words and, so far, only use bubbles in which the middle sequence has a maximum length of three[9] words. The notion of contexts from bubbles is clarified with the next example.

```
L2  L1  X1 X2 X3  R1  R2  R3  R4
L2  L1  __ __ __  R1  R2  R3  R4
```

For such a case, we consider `[L1, L2]` as the left context, `[R1, R2, R3]` as the right context, and `[X1, X2, X3]` as the aligned middle sequence. Such an example is represented with a Prolog term with arity 5 (`bub/5`) in the following manner:

```
bub(ID, t(3,0), [L1,L2],
    [X1,X2,X3]--->[],
    [R1,R2,R3]).
```

The `ID` is the identifier of the sequence instance, `t/2` defines the "transformation dimension", in this case from 3 words to 0. The third and fifth arguments are lists with the left and right contexts, respectively, and the fourth argument contains the list with the elements deleted from the middle sequence. It is important to point out that every L_i, X_i and R_i are structures with 3 elements such as `word/POS/Chunk`. For example, the word `president` would be represented by the expanded structure `president/nn/np`.

4.3 Feature Space

As mentioned previously, with an ILP system, and in particular with *Aleph*, the set of attributes is defined through a set of conditions, expressed in the form of predicates. These predicates are the building blocks that will be employed to construct rules, during the induction process. Hence, our attribute search space is defined using Prolog predicates, which define the complete set of possibilities for rule body construction. In our problem, we let the induction engine seek generalization conditions for the bubble main regions (left, middle, and right). Each condition may be from one of the four types: dimensional, lexical, POS, and chunk. Dimensional conditions simply express the aligned sequence transformation dimensionality. Lexical conditions impose a fixed position to match a given word. The POS condition is similar to the lexical one, but more general, as the position must match a specific part-of-speech tag. Likely, chunk conditions bind a region to be equal to a particular chunk type. For example, by looking

[9]They represent 83.47% from the total number of extracted bubbles.

at Figure 2, the attentive reader may have noticed that these three conditions are present in rule 7. In terms of *Aleph* declaration mode, these conditions are defined as follows.

```
:- modeh(1,rule(+bub)).

:- modeb(1,transfdim(+bub,n(#nat,#nat))).
:- modeb(3,chunk(+bub,#side,#chk)).
:- modeb(*,inx(+bub,#side,#k,#tword)).

:- determination(rule/1,transfdim/2).
:- determination(rule/1,chunk/3).
:- determination(rule/1,inx/4).
```

The `inx/4` predicate defines lexical and POS type conditions, the `chunk/3` predicate defines chunking conditions and the `transfdim/2` predicate defines the transformation dimensionality, which is in the form `transfdim(N,0)` with `N>0`, according to the kind of bubbles we are working with.

4.4 The Rule Value Function

The *Aleph* system implements many different evaluation[10] functions which guide the theory search process, allowing the basic procedure for theory construction to be altered. In order to better fit to our problem, we define a new evaluation function calculated as the geometrical mean between the coverage percentage and the rule size value, as shown in Equation 11 where R is the candidate rule and $Cov(R)$ is the proportion of positive instances covered by R and the $LV(\bullet)$ function defines the rule value in terms of its length, returning a value in the $[0, 1]$ interval.

$$Value(R) = \sqrt{Cov(R) \times LV(R)} \qquad (11)$$

The $Value(\bullet)$ function guides the induction process, by preferring not too general rules having maximum possible coverage value. As shown in Figure 7, the $Value(\bullet)$ function gives preferences to rules with 3, 4 and 5 literals.

5 Results

The automatic evaluation of a system is always the best way to do it, due to its objectivity and scalability. However, in many cases it is unfeasible for several practical reasons, like the unavailability of data or the difficulty to prepare an appropriate

[10]In the *Aleph* terminology, this function is named as the "cost" function, despite the fact that it really computes the value in the sense that the grater the value, the more likely it is to be chosen.

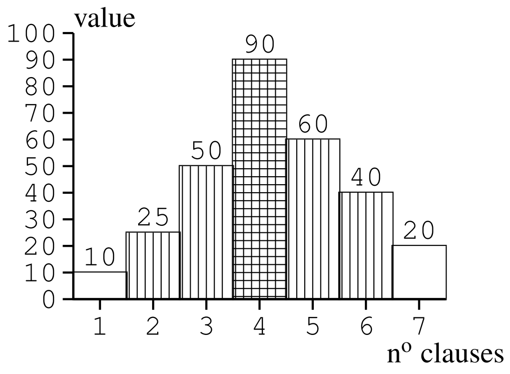

Figure 7: Rule length value function

dataset. Some supervised learning approach use manually labeled test sets to evaluated their systems. However, these are small test sets, for example, (Knight & Marcu, 2002) use a set of 1035 sentences to train the system and only 32 sentences to test it, which is a quite small test set. As a consequence, it is also important to propose more through evaluation. In order to assess as clearly as possible the performance of our methodology on large datasets, we propose a set of qualitative and quantitative evaluations based on three different measures: Utility, Ngram simplification and Correctness.

5.1 Evaluation

A relevant issue, not very commonly discussed, is the *Utility* of a learned theory. In real life problems, people may be more interested in the volume of data processed than the quality of the results. Maybe, between a system which is 90% precise and processes only 10% of data, and a system with 70% precision, processing 50% of data, the user would prefer the last one. The Utility may be a stronger than the Recall measure, used for the evaluation of supervised learning systems, because the later measures how many instances were well identified or processed from the test set only, and the former takes into account the whole universe. For example, in a sentence compression system, it is important to know how many sentences would be compressed, from the whole possible set of sentences encountered in electronic news papers, or in classical literature books, or both. This is what we mean here by Utility.

The *Ngram-Simplification* methodology is an automatic extrinsic test, performed to perceive how much a given sentence reduction ruleset would simplify sentences in terms of syntactical complexity. The answer is not obvious at first sight, because even smaller sentences can contain more improbable syntactical subsequences than their uncompressed versions. To evaluate the syntactical complexity of a sentence, we use a $4-gram$ model and compute a relative[11] sequence probability as defined in Equation 12 where $\vec{W} = [t_1, t_2, ..., t_m]$ is the sequence of part-of-speech tags for a given sentence with size m.

$$P\{\vec{W}\} = \Big(\prod_{k=n}^{m-n} P\{t_k \mid t_{k-1}, ..., t_{k-n}\}\Big)^{\frac{1}{m}} \qquad (12)$$

The third evaluation is qualitative. We measure the quality of the learned rules when applied to sentence reduction. The objective is to assess how correct is the application of the reduction rules. This evaluation was made through manual annotation for a statistically representative random sample of compressed sentences. A human judged the adequacy and *Correctness* of each compression rule to a given sentence segment, in a scale from 1 to 5, where 1 means that it is absolutely incorrect and inadequate, and 5 that the compression rule fits perfectly to the situation (sentence) being analyzed.

To perform our evaluation, a sample of 300 sentences were randomly extracted, where at least one compression rule had been applied. This evaluation set may be subdivided into three subsets, where 100 instances came from rules with $Z_{(X)} = 1$ (**BD1**), 100 from rules with $Z_{(X)} = 2$ (**BD2**), and the other 100 from rules with $Z_{(X)} = 3$ (**BD3**). Another random sample, also with 100 cases has been extracted to evaluate our base-line (**BL**) which consists in the direct application of the bubble set to make compressions. This means that no learning process is performed. Instead, we store the complete bubble set as if they were rules by themselves (in the same manner as (Le Nguyen & Ho, 2004) do).

Table 1 compiles the comparative results for Correctness, Precision, Utility and Ngram-simplification for all datasets. In particular, Ngram-simplification in percentage is the proportion of test cases where $P\{reduced(\vec{W})\} \geq P\{\vec{W}\}$.

Table 1 provides evidence of the improvement achieved with the induction rules, in comparison with the base line, on each test parameter: Correctness, Utility and Ngram-simplification. Considering the three experiences, **BD1**, **BD2**, and **BD3**, as a unique evaluation run, we obtained a mean Correctness quality of 3.867 (i.e. 77.33% Precision), a mean Utility of 48.45%, and a mean Ngram-simplification equal to 89.53%, which are significantly better than the base line.

[11]Because it is raised to the inverse power of m, which is the number of words in the sentence.

Parameter	**BL**	**BD1**	**BD2**	**BD3**
Correctness:	2.93	3.56	4.03	4.01
Precision:	58.60%	71.20%	80.60%	80.20%
Utility:	8.65%	32.67%	85.72%	26.86%
NG-Simpl:	47.39%	89.33%	90.03%	89.23%

Table 1: Results with Four Evaluation Parameters.

Moreover, best results overall are obtained for **BD2** with 80.6% Precision, 85.72% Utility and 90.03% Ngram-simplification which means that we can expect a reduction of two words with high quality for a great number of sentences. In particular, Figure 2 shows examples of learned rules.

5.2 Time Complexity

In the earlier[12] days of ILP, the computation time spent by their systems was a serious difficult obstacle, disabling its implementation for real life problems. However, nowadays these time efficiency issues have been overcome, opening a wide range of application possibilities, for many problems, from Biology to Natural Language Processing. The graph in figure 8, shows that even with considerable big datasets, our learning system (based on *Aleph*) evidences acceptable feasible computation time.

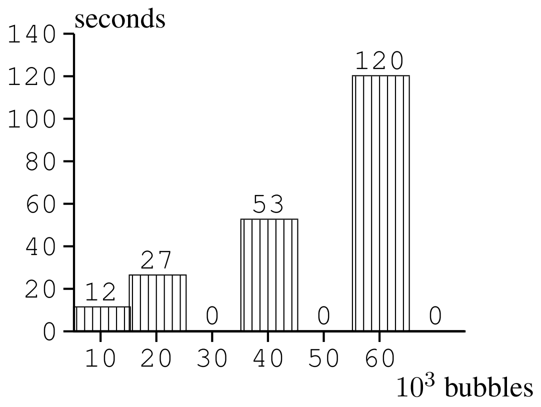

Figure 8: Time spent during the induction process, for datasets with size expressed in thousands of bubbles.

To give an idea about the size of an induced rule set, and taking as an example the learned rules

[12]In the 1990-2000 decade.

with $Z_{(X)} = 2$, these were learned from a dataset containing 37271 $t(2,0)$ bubbles, and in the final 5806 sentence reduction rules were produced.

6 Conclusion and Future Directions

Sentence Compression is an active research topic, where several relevant contributions have recently been proposed. However, we believe that many milestones still need to be reached. In this paper, we propose a new framework in the form of a pipeline, which processes huge sets of web news articles and retrieves compression rules in an unsupervised way. For that purpose, we extract and align paraphrases, explore and select specific text characteristics called bubbles and finally induce a set of logical rules for sentence reduction in a real-world environment. Although we have only considered bubbles having $Z_{(X)} \leq 3$, a sentence may have a compression length greater than this value, since several compression rules may be applied to a single sentence.

Our results evidence good practical applicability, both in terms of Utility, Precision and Ngram-simplification. In particular, we assess results up to 80.6% Precision, 85.72% Utility and 90.03% Ngram-simplification for reduction rules of two word length. Moreover, results were compared to a base line set of rules produced without learning and the difference reaches a maximum improvement using Inductive Logic Programming of 22%.

Acknowledgments

This work was supported by the VIPACCESS project - *Ubiquitous Web Access for Visually Impaired People*. Funding Agency: *Fundação para a Ciência e a Tecnologia* (Portugal). Reference: PTDC/PLP/72142/2006.

References

Barzilay R. and Lee L.. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003: Main Proceedings*, pages 16–23.

Camacho R. 1994. Learning stage transition rules with Indlog. *Gesellschaft für Mathematik und Datenverarbeitung MBH.*, Volume 237 of GMD- Studien, pp. 273-290.

Clarke J., and Lapata M. 2006. Constraint-based Sentence Compression: An Integer Programming Approach. *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics.*

Cordeiro J. and Dias G. and Cleuziou G. 2007. Biology Based Alignments of Paraphrases for Sentence Compression. *In Proceedings of the Workshop on Textual Entailment and Paraphrasing* (ACL-PASCAL / ACL2007), Prague, Czech Republic.

Cordeiro J. and Dias G. and Brazdir P. October 2007. New Functions for Unsupervised Asymmetrical Paraphrase Detection. *In Journal of Software.*, Volume:2, Issue:4, Page(s): 12-23. Academy Publisher. Finland. ISSN: 1796-217X.

Cordeiro J. and Dias G. and Brazdir P. August 2009. *In Proceedings of the Workshop on Language Generation and Summarisation (UCNLG+Sum)* (ACL-IJCNLP / ACL2009), Singapore.

Dolan W.B. and Quirck C. and Brockett C. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of 20th International Conference on Computational Linguistics (COLING 2004).*

Knight K. and Marcu D. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91-107.

Muggleton S. 1991. *Inductive Logic Programming. New Generation Computing*, 8 (4):295-318.

Muggleton S. 1996. *Learning from positive data. Proceedings of the Sixth International Workshop on Inductive Logic Programming (ILP-96), LNAI 1314*, Berlin, 1996. Springer-Verlag.

Muggleton S. 1999. *Inductive logic programming: Issues, results and the challenge of learning language in logic. Artificial Intelligence*, 114 (1-2), 283?296.

Le Nguyen M., Horiguchi S., A. S., and Ho B. T. 2004. Example-based sentence reduction using the hidden markov model. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(2):146-158.

Needleman SB, Wunsch CD. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48 (3): 443–53.

Quinlan J. R. 1990. *Learning Logical Deinitions from Relations. Machine Learning.*, 5 (3), 239-266. 33, 39, 41.

Smith TF, Waterman MS. 1981. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147: 195–197.

Srinivasan A. 2000. *The Aleph Manual*, Technical Report. Computing Laboratory, Oxford University, UK.

Turner J, Charniak E. 2005. Supervised and Unsupervised Learning for Sentence Compression. Proceedings of the 43rd Annual Meeting of the ACL, pages 290-297.