

# A Corpus Based Morphological Analyzer for Unvocalized Modern Hebrew

Alon Itai and Erel Segal

Department of Computer Science  
Technion—Israel Institute of Technology, Haifa, Israel

## Abstract

Most words in Modern Hebrew texts are morphologically ambiguous. We describe a method for finding the correct morphological analysis of each word in a Modern Hebrew text. The program first uses a small tagged corpus to estimate the probability of each possible analysis of each word regardless of its context and chooses the most probable analysis. It then applies automatically learned rules to correct the analysis of each word according to its neighbors. Finally, it uses a simple syntactical analyzer to further correct the analysis, thus combining statistical methods with rule-based syntactic analysis. It is shown that this combination greatly improves the accuracy of the morphological analysis—achieving up to 96.2% accuracy.

## 1. Introduction

Most words in Modern Hebrew (henceforth, Hebrew) script are morphologically ambiguous. This is due to the rich morphology of the Hebrew language and the inadequacy of the common way in which Hebrew is written. Morphological disambiguation is a must for many applications, such as spellers, search engines and machine translation. When used as a front-end of a syntactic parser, it helps to reduce the number of incorrect parses (Sima'an, 2002).

Morphological analysis partitions a word token into morphemes and features. The morphemes consists of the lexical lemma, short words such as determiners, prepositions and conjunctions that are prepended to the word, suffixes for possessives and object clitics. The linguistic features mark part-of-speech (POS), tense, person etc. Following Kempe, we partition this data to a lexical lemma and a *tag*. For example, the first analysis of the ambiguous word token \$QMTI (שקמתי) is “my sycamore” whose lemma is \$QMH and tag is [*noun: definite, feminine: s + possessive: 1sg*]. The second analysis of the same word token is “that I got up”; its lemma is QM and its tag is [*connective + verb: 1sg: past*]. The morphological analysis consists of determining the tag and the lemma. Thus, determining the correct tags in context is similar to POS tagging in English.

POS tagging in English has been successfully attacked by corpus based methods. Thus we hoped to adapt successful POS tagging methodologies to

the morphological analysis of Hebrew. There are two basic approaches: Markov Models (Church 1988, DeRose 1988) and acquired rule-based systems (Brill, 1995). Markov Model based POS tagging methods were not applicable, since such methods require large tagged corpora for training. Weischedel et al (1994) use a tagged corpus of 64,000 words, which is the smallest corpus we found in the literature for HMM tagging, but is still very big. Such corpora do not yet exist for Hebrew. We preferred, therefore, to adapt Brill's rule based method, which seems to require a smaller training corpus. Brill's method starts with assigning to each word its most probable morphological tag, and then applies a series of “transformation rules”. These rules are automatically acquired in advance from a modestly sized training corpus.

In this work, we find the correct morphological analysis in context by combining probabilistic methods with syntactic analysis. The solution consists of three consecutive stages:

(a) The word stage: In this stage we find all possible morphological analyses of each word in the analyzed text. Then we approximate, for each possible analysis, the probability that it is the correct analysis, independent of the context of the word. For this purpose, we use a large unanalyzed training corpus. After approximating the probabilities, we assign each word the analysis with the highest approximated probability (this stage is inspired by Levinger et al, 1995).

(b) The pair stage: In this stage we use transformation rules, which correct the analysis of a word according to its immediate neighbors. The transformation rules are learned automatically from a training corpus (this stage is based on Brill, 1995).

(c) The sentence stage: In this stage we use a rudimentary syntactical parser to evaluate different alternatives for the analysis of whole sentences. We use dynamic programming to find the analysis which best matches both the syntactical information obtained from the syntactical analysis and the probabilistic information obtained from the previous two stages.

The data for the first two stages are acquired automatically, while the sentence stage uses a manually created parser.

These three stages yield a morphological analysis which is correct for about 96% of the word tokens, thus approaching results reported for English probabilistic POS tagging. Our method uses a very small training corpus – only 4900 words, similar in size to the corpus used by Brill and much smaller than the million-word corpora used for HMM based POS tagging of English. The results show that combining probabilistic methods with syntactic information improves the accuracy of morphological analysis.

In addition to solving a practical problem of Modern Hebrew and other scripts that lack vocalization (such as Arabic, Farsi), we show how several learning methods can be combined to solve a problem, which cannot be solved by any one of the methods alone.

Because of space limitations, this description is very terse, we had no space to give algorithms and show in detail the statistical analyses.

### 1.1. Previous Work

Both academic and commercial systems have attempted to attack the problem posed by Hebrew morphology. The Rav Millim (Choueka, 2002) commercial system provided a morphological analyzer and disambiguator. Within a Machine Translation project, IBM Haifa Scientific Center provided a morphological analyzer (Ben-Tur et al., 1992) that was later used in several commercial products. Since these sources are proprietary, we

used Segal's publicly available morphological analyzer (Segal 2001).

Other works attempted to find the correct analysis in context. Choueka and Luisignan (Choueka and Lusingnan, 1985) proposed to consider the immediate context of a word and to take advantage of the observation that quite often if a word appears more than once in the same document the same analysis will be the correct analysis throughout the document. Orly Albeck (1992) attempted to mimic the way humans analyze texts by manually constructing rules that would allow finding the right analysis without backtracking. Levinger (1992) gathered statistics to find the probability of each word, and then used hand crafted rules to rule out ungrammatical analyses. Carmel and Maarek (1999) used statistics on tags similar to ours, to partially disambiguate words and then indexed the disambiguated text for use in a search engine. The first stage of our work, the word stage, was based on Levinger (1995), but like Carmel (1999) used the morphological tags independently of the lemmas. Ornan and Katz built a disambiguation system for Hebrew based on the phonemic script and handcrafted semantic clues (Ornan 1994).

### 1.2 The basic morphological analyzer

A basic morphological analyzer is a function that inputs a word token and returns the set of all its possible morphological analyses. The analyzer we used supplies all the morphological information, except for the object clitics. We found only two object clitics in a 4900 word corpus of a Hebrew newspaper, so we concluded that adding the object clitics to the analysis won't add much to its coverage, while substantially increasing the ambiguity. Other analyzers, such as Rav-Milim, identify the object clitic in some but not all of the words.

In this work, we didn't intend to tackle the problem of the several standards of unvocalized orthography, so we used a conservative analyzer that identified only "full-script" unvocalized words (ktiv male). However, the same methods can easily be applied to other standards.

## 2. The word stage

We followed Levinger et al. (Levinger 1995) and used a variant of their algorithm, the similar words

algorithm, to find the probability of each analysis, regardless of context. As in Levinger et al. the probability of a lemma  $m$  of a word  $w$ , was estimated by looking at other words that contained  $m$  but differed from  $w$  in its tags. The frequency of the new words was found by counting their occurrences in an untagged corpus of 10 million words from the daily Hebrew press.

The main difference from Levinger was that to overcome the sparseness problem of the data, we followed Carmel (1992) and assumed that the occurrences of the tags are statistically independent and estimated the probability of each tag independently. The probability of an analysis was derived by multiplying the probability of the tag by that of the lemma. Even though the assumption is not always valid (Altman 2002), in most cases this procedure correctly ranked the analyses.

### 3. The pair stage

#### 3.1 Using transformation rules to improve the analysis

The concept of rules was introduced by Brill (1995), who first used acquired transformation rules to build a rule-based POS tagger. He argued that transformation rules have several advantages over other context-dependent POS tagging methods (such as Markov models):

(a) The transformation-rule method keeps only the most relevant data. This both saves a lot of memory space and enables the development of more efficient learning algorithms.

(b) Transformation rules can be acquired using a relatively small training corpus.

We too use transformation rules, but in contrast to Brill, our transformation rules do not automatically change the analyses of the matching words. In order to use the probabilistic information gathered in the word stage, we assign each possible analysis of each word a morphological score. The score of each analysis is initialized to the probability as determined at the word stage. The transformation rules modify the scores of the analyses. The modified scores can be used to select a single analysis for each word (that with the highest score), or used as an input to a higher level analyzer (such as the syntactic analyzer to be described below).

A transformation rule operates on a pair of adjacent words. The general syntax of a transformation rule is:

**pattern<sub>1</sub> pattern<sub>2</sub> [agreement] →  
newpattern<sub>1</sub>(+inc<sub>1</sub>) newpattern<sub>2</sub>(+inc<sub>2</sub>)  
[agreement]**

Both the left-hand side and a right-hand side of a rule contain two *analysis patterns* and an optional *agreement pattern*. An analysis-pattern is any pattern that matches the tag of a single word. An agreement pattern is a pattern that indicates how two adjacent tags agree, for example "agreeing-by-gender", "agreeing-by-gender-and-number", etc.

A rule comes into effect only for pairs of adjacent tags, where the first tag matches "**pattern<sub>1</sub>**", the second tag matches "**pattern<sub>2</sub>**", and the two tags agree according to "**agreement**".

Here is an example of a transformation rule:

proper-name noun → proper-name(+0) verb  
(+0.5) agreeing-in-gender

Its meaning is as follows: Let  $w_1$ ,  $w_2$  be two adjacent words

**If** the POS of the current tag of  $w_1$  is a proper-noun and the POS of the current tag of  $w_2$  is a noun

**and**  $w_2$  has an analysis as a verb that matches  $w_1$  by gender and number,

**then** add 0.5 to the morphological score of  $w_2$  as a verb, and normalize the scores .

Consider the combination "YWSP &DR" (רֹסֵף עֶרֶד). The word "&DR" has two possible analyses: one as a masculine noun (= herd) and the other as a verb (masculine past-tense 3sg; = hoed). Suppose our analyzer found, in the word stage, that the most probable analysis of the word "YWSP" is a masculine proper name (=Joseph), and the most probable analysis of the word "&DR" is a noun (= herd). The current analysis of this combination is "Joseph herd", which is most unlikely. However, this combination of analyses matches the first part of the transformation rule: the current analysis of  $w_1$  is a proper noun and the current analysis of  $w_2$  is a noun. Moreover,  $w_2$  has an analysis that matches the second part of the rule: a verb that matches  $w_1$  by gender. Therefore, the rule will add 0.5 to the morphological score of the other analysis

of  $w_2$ . If the difference between the scores of the two analyses was less than 0.5 – the highest-scored analysis of  $w_2$  will now be the verb, so that the analysis of the entire combination will be “Joseph hoed”. Had the difference between the scores been greater than 0.5 – the analysis would not have changed.

Rules can also depend on the lemma of the words, eg.:

Rules follow the following template

$T_1 T_2 \rightarrow T_3 T_4$  **agreement-pattern inc**

Where  $T_i$  may be either a POS or a specific word, the agreement pattern is any combination of agreements of Hebrew (gender, number, definiteness). The increment **inc** is a positive or negative number, that is added to the morphological score. For example,

'HWA' noun  $\rightarrow$  'HWA'(+0) verb agreeing-in-person-gender-and-number(+0.5)

### 3.2. Acquiring the transformation rules

Transformation rules are acquired automatically using an analyzed training corpus. The learning algorithm uses the following input:

(a) For each word in the training corpus: the set of its analyses, and the morphological score of each analysis.

(b) The correct analysis of each word in the training corpus.

The output of the algorithm is an ordered list of transformation rules.

The learning algorithm proceeds as follows:

- a. **(Initialization)**: Assign to each word its most probable analysis.
- b. **(Transformation rule generation)**: loop over all incorrectly tagged words in the corpus. Generate all transformation rules that correct the error (**inc** is yet undermined).
- c. **(Determining inc)** For each correction, create a rule whose increments is the minimum required to perform the correction.
- d. **(Transformation rule evaluation)**: loop over the candidate transformation rules and retain the rule that corrects the maximum number of errors, while causing the least damage.
- e. Repeat the entire process until the net gain of all rules is negative.

The process terminates, since in each iteration the number of errors in the training corpus decreases. The worst-case complexity of the algorithm is  $O(N^3)$ , where  $N$  is the size of the training corpus.

## 4. The sentence stage

The aim of this stage is to improve the accuracy of the analysis using syntactic information. A correct analysis must correspond to a syntactically correct sentence. We therefore try to syntactically parse the sentence (actually the tags of the sentence). If the parse fails, we would like to conclude that the proposed analysis is incorrect and try another morphological analysis. However, since no syntactic parser is perfect, we do not reject sentences that failed to parse. We use the syntactic grammaticality, estimated by the syntactic parser, as one of two measures for the correctness of the analysis, and combine this with the score that results from the pair phase.

### 4.1 The grammar

Our syntactic parser uses a handcrafted grammar of about 150 rules. The rules attempt to simplify the sentence, for example, the rule

**noun  $\rightarrow$  noun adjective**

[agree in number and gender],

reduces two tokens into one. Additional rules account for entire sentences, such as

**sentence  $\rightarrow$  noun-phrase verb-phrase**

and recursive structures, such as

**noun  $\rightarrow$  noun connective sentence .**

For an input sentence  $w_1, \dots, w_n$ , let  $T_i = \{t_{i1}, \dots, t_{ik}\}$  be the set of tags of  $w_i$ , and  $s_{im}$  be the score of  $t_{im}$  as determined by the previous stages. If we assume that the scores are probabilities and that the probability of choosing tags for words are statistically independent; i.e.,

$P(\text{tag}(w_1) = t_{1m_1} \text{ and } \text{tag}(w_2) = t_{2m_2} \dots \text{ and}$

$\text{tag}(w_n) = t_{nm_n})$

$= P(\text{tag}(w_1) = t_{1m_1}) P(\text{tag}(w_2) = t_{2m_2}) \dots$

$P(\text{tag}(w_n) = t_{nm_n})$

$= s_{1m_1} s_{2m_2} \dots s_{nm_n}$ ,

maximizing the product is equivalent to finding the most probable tag sequence. Thus we wish to

maximize the product under the constraint that the sequence is a syntactically correct sentence (or at least nearly correct).

## 4.2 The algorithm.

### 4.2.1 Dynamic Programming

Our algorithm uses dynamic programming to determine the score of partial parses. For a nonterminal  $A$ , let a  $Table[i,j,A]$  be the maximum score of all parses  $A \xrightarrow{*} w_i \cdots w_j$ . (If  $w_i \cdots w_j$  cannot be derived from  $A$ , then  $Table[i,j,A] = 0$ .) If we consider the scores as probabilities then  $Table[i,j,A]$  is the probability of the best parse that derives  $w_i \cdots w_j$  from  $A$ .

$Table$  is computed by increasing value of  $\ell = j - i$ :

$\ell = 0$ :

$$Table[i,i,A] = \max \{s_{im} : A \rightarrow t_{im} \in G \text{ and } t_{im} \in T_i\}.$$

To compute  $Table[i,i,A]$  we check for all rules  $A \rightarrow t$  if  $t \in T_i$ . The time is linear in  $g_A$ , the number of grammar rules whose left hand side is  $A$ . Thus, computing  $Table[i,i,\cdot]$  is linear in  $g$ , the size of the grammar  $G$ . Hence, computing the first row requires  $O(ng)$  time.

$\ell > 0$ :

$$Table[i,j,A] = \max_{\substack{A \rightarrow BC \in G \\ i \leq k < j}} \{Table[i,k,B] \times Table[k+1,j,C]\}$$

The time to find the maximum is  $(j-i+1)g_A$ .

Thus, computing  $Table[i,j,\cdot]$  for all  $i,j, j-i=\ell$  requires time  $O(\ell(n-\ell)g) = O(n^2g)$ , and for all

$1 \leq \ell \leq n$  the time is  $O(n^3g)$ .

### 4.2.2 Parsing the sentence

Ideally, the score should be  $T[1,n,S]$ . However, no parser is perfect and our rudimentary parser is no exception. Since a properly analyzed sentence should consist of simple sentences connected by connectives, we try to cover the sentence with a minimum number of components. In other words, we look for  $r$  and  $k_1 < k_2 < \cdots < k_r$  such that

$$Table[1,k_1,A_1] + Table[k_1+1,k_2,A_2] + \dots + Table[k_r+1,n,A_r] - f(r)$$

is maximum. The function  $f$  is monotonically increasing, reflecting the cost of having  $r$  components. The rationale is that we should get a bonus for choosing the best tags and pay a fine for failing to parse.

The function  $f$  should have been determined by Machine learning techniques, but we assumed that  $f(r) = \alpha r$ . Under this assumption, finding the “best parse” is equivalent to finding a shortest path in a directed graph from 1 to  $n$  in a graph whose vertex set is  $\{1, \dots, n\}$  and  $d(i,j) = \alpha - \max_A \{Table[i,j,A]\}$ . If all the distances are nonnegative, we can apply Dijkstra’s algorithm whose complexity is  $O(n^3)$ .

### 4.2.3 Time complexity

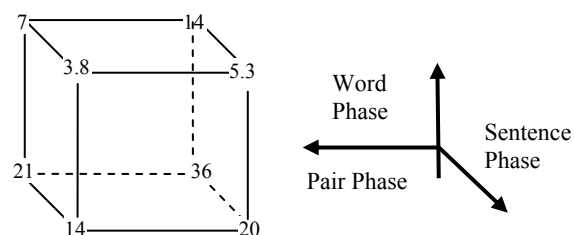
The complexity of the entire algorithm is dominated by the dynamic programming step which requires  $O(n^3g)$  time.

## 5. Evaluation

To test the algorithm we used an analyzed corpus of 5361 word tokens, which contained 16 articles of various subjects from a Hebrew daily newspaper. 468 word tokens were used for testing and the rest for training. The results are summarized in Table 1.

Word Stage	Pair Stage	Sentence Stage	Error (%)
No	No	No	36.0
Yes	No	No	14.0
No	Yes	No	21.0
Yes	Yes	No	7.0
No	No	Yes	20.0
Yes	No	Yes	5.3
No	Yes	Yes	14.0
Yes	Yes	Yes	3.8

**Table 1:** The percentage of error when using each method separately and in combination with other methods.



**Figure 1:** A graphical representation of Table 1

The first line (No, No, No) assumed that all analyses are equiprobable. The error percent, 36% , reflects the expected number of analyses of a word in the test data ( $1/0.36 \approx 2.8$ ). The second line (Yes, No, No) considers only the word stage, and it is equivalent to Levinger (1995). Subsequent lines show using different combinations of stages. The word stage seems most essential – leaving it out most degrades the performance. The pair stage and the sentence stage both use syntactic structure. The pair stage improved the scores of the tags of words based on their neighbors, while the sentence stage utilized a broader scope. It seems that using both yielded the best results; however, we were not able to show the statistical significance of this improvement. (We tested the significance of the difference of proportions. For this pair the confidence level was 86%.) All other pairs of results showed statistical significance with confidence level above 95%.

To test the effect of the corpus size on the pair stage, we performed two experiments, each with a different test article:

- Article A with 469 word tokens (which leaves 4892 word tokens in the training corpus),
- Article B with 764 word tokens (which leaves 4597 word tokens in the training corpus),

In order to examine how the size of the training corpus affects the number of transformation rules learned and the final accuracy, we conducted  $k$ -way cross and took the average. The results are shown in Appendix B. A statistical analysis revealed that with 95% confidence the error rate at most 8%.

The error-rate graphs are quite flat even for this small training corpus. Perhaps it is possible to conclude that using a larger training corpus won't make the results much better. Right now, however, we cannot verify this conclusion because we don't have a much larger tagged corpus. Such a corpus is in the process of being prepared (Sima'am 2001).

## 7. Conclusions and further research

The research shows that corpus based methods are effective for choosing the correct morphological analysis in context. However, there is still a considerable gap between our system and human readers. From the error analysis, we see that our system treats all unknown words as proper nouns, and does not recognize idioms. At least two more

“expert systems” need be incorporated: (a) a recognizer for proper nouns, and (b) a recognizer of idioms.

Furthermore, the sentence stage is not automatic, and is not sufficiently robust. We plan on attacking this problem in several ways. Currently, Sima'an et al (2001) is creating a tree-bank that will enable to automatically learn a grammar for Hebrew. However, even when complete, parsing will be slow. Another vein of research is to follow Abney (1996) and construct a finite state cascade parser. Even though such parsers do not provide full coverage, they are very fast and may be sufficient for the purpose of morphological disambiguation.

## Bibliography

- Steven Abney. 1996. Partial Parsing via Finite-State Cascades. In *J. of Natural Language Engineering*, 2(4), pp. 337-344.
- Orly Albeck. 1992. Formal analysis by a restriction grammar on one of the stages of Modern Hebrew. Computerized analysis of Hebrew words. In *Israel Ministry of Science and Technology Symposium*. (In Hebrew.)
- Alon Altman. 2002. Private communication.
- Esther Ben-Tur, Aviela Angel, Danit Ben-Ari, and Alon Lavie. 1992. Computerized analysis of Hebrew words. In *Israel Ministry of Science and Technology Symposium*. (In Hebrew.)
- Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. In *Computational Linguistics*, 21, pp. 543-565.
- David Carmel and Yoëlle Maarek. 1999. Morphological disambiguation for Hebrew search systems. In *Next Generation Information Technologies and Systems, NGITS '99*, Springer LNCS 1649, pp. 312-325.
- Yaacov Choueka and S. Lusin. 1985. Disambiguation by short context. In *Computers and the Humanities*, 19(3).
- Yaacov Choueka. 2002. Rav Millim. Technical report, The Center for Educational Technology in Israel. (<http://www.cet.ac.il/rav-milim/>)
- Kenneth W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *ANLP*, 2, pp. 136-143.

Steven J. DeRose. 1988. Grammatical category disambiguation by statistical optimization, In *Computational Linguistics*, 14, pp. 31-39.

ISO. 1999. "Information and documentation – Conversion of Hebrew characters into Latin characters – Part 3: Phonemic Conversion, ISO/FDIS 259-3: (E).

André Kempe Probabilistic parsing with feature structures.

Moshe Levinger, Uzzi Ornan and Alon Itai. 1995. Morphological disambiguation in Hebrew using a priori probabilities, *Computational Linguistics* 21, pp. 383-404.

Uzzi Ornan and Michael Katz. 1994. A new program for Hebrew index based on the phonemic script, *Technical Report #LCL 94-7*,

*Laboratory for Computational Linguistics, CS Dept., Technion, Haifa, Israel.*

Erel Segal. 2001. A Hebrew morphological analyzer (includes free source code) <http://come.to/balshanut> or <http://www.cs.technion.ac.il/~erelsgl/bxi/hmntx/teud.html>

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman and Noa Nativ. 2001. Building a Tree-Bank of Modern Hebrew Text. In *Traitment Automatique des Langues*, 42, pp. 347-380.

Ralph M. Weischedel, Marie Meteer, Richard L. Schwartz, Lance Ramshaw and Jeff Palmucci. 1994. Coping with Ambiguity and Unknown Words through Probabilistic Models. In *Computational Linguistics*. 19(2), pp. 359-382.

## Appendix A: The Hebrew-Latin transliteration

The choice of letters follows the ISO standard of phonemic script (ISO 1999). Note that this is not a phonetic transcription. The vowels 'a' and 'e', which are usually not represented in the Hebrew unvocalized script, are also not represented in our Latin transliteration. For example, the word: שלג which is pronounced "sheleg", is transliterated: \$LG.

## Appendix B: The results of the pair-stage experiment

The first graph shows how the number of rules grows with the size of the corpus. If we had a larger corpus then we would have need to trim the less effective rules to avoid overfitting.

The second graph shows how the error rate decreases with the size of the training corpus.

Heb. letter	Heb. name	Latin	Heb. letter	Heb. name	Latin
א	Alef	A	ל	Lamed	L
ב	Bet	B	מ,ם	Mem	M
ג	Gimm el	G	נ,ן	Nun	N
ד	Dalet	D	ס	Samek	S
ה	Hei	H	ע	Ayn	&
ו	Waw	W	פ,ף	Pei	P
ז	Zayn	Z	צ,ץ	Tsadiq	C
ח	Xet	X	ק	Quf	Q
ט	Tet	@	ר	Reish	R
י	Yud	I	ש,שׁ	Shin	\$
כ,ך	Kaf	K	ת	Taw	T

The third graph shows a better picture by neutralizing the errors detected at the word stage, i.e. it shows the percent of errors after the word phase minus the percent of errors after the pair stage.

article A : 469 words	# words for training [a]	# learned rules [a]	initial # errors [a]	initial % errors [a]	final # errors [a]	final % errors [a]	effect of rules [a]
	0	0	80	17.1%	80	17.1%	0.0%
	701	22	79	16.8%	51	10.9%	6.0%
	1073	30	77	16.4%	42	9.0%	7.5%
	2105	48	81	17.3%	42	9.0%	8.3%
	2929	63	79	16.8%	39	8.3%	8.5%
	3851	78	70	14.9%	39	8.3%	6.6%
	4892	93	68	14.5%	29	6.2%	8.3%
article B : 764 words	# words for training [b]	# learned rules [b]	initial # errors [b]	initial % errors [b]	final # errors [b]	final % errors [b]	effect of rules [b]
	0	0	140	18.3%	140	18.3%	0.0%
	468	14	129	16.9%	90	11.8%	5.1%
	695	18	129	16.9%	86	11.3%	5.6%
	1713	45	130	17.0%	73	9.6%	7.5%
	2640	53	128	16.8%	71	9.3%	7.5%
	3562	76	131	17.1%	64	8.4%	8.8%
	4597	90	126	16.5%	53	6.9%	9.6%

