How Symbolic Learning Can Help Statistical Learning (and vice versa)

Isabelle Tellier
Lattice / Lattice - UMR 8094
Lattice / 1 rue Maurice Arnoux
Lattice / 92120 Montrouge
isabelle.tellier@univ-paris3.fr

Yoann Dupont
Lattice / Lattice - UMR 8094
Lattice / 1 rue Maurice Arnoux
Lattice / 92120 Montrouge
yoa.dupont@gmail.com

Abstract

We describe in this paper how different learning strategies can be applied on the same NLP task, namely chunking. The reference corpus is extracted from the French Treebank, the symbolic learning strategy used is grammatical inference and the statistical one is CRFs (Conditional Random Fields). As expected, the symbolic approach allows readability but is less effective than the statistical one. We then propose two distinct ways to combine both approaches and show that in both cases they benefit from one another.

1 Introduction

Supervised machine learning approaches, especially when they have access to huge amounts of data, have now extensively proved their effectiveness for a lot of text mining tasks like text classification, sentence annotation and information extraction. Most effective learning approaches rely on a theoretical background which is either optimization (SVM), statistics (Naive Bayes) or both (HMMs, MaxEnt models, CRFs). But, however effective they may be, the main drawback of these techniques is that they usually do not provide any human-readable model.

There also exists other branches of Machine Learning, referred to as *symbolic*, whose particularity is to provide a more human-readable output. This is the case of decision trees, Inductive Logic Programming (ILP) or Grammatical Inference (GI in the following). The latter is our main interest here. It can be defined as the study of how it is possible to automatically learn a formal grammar or any other device able to represent a *language* (such as an automaton, a regular expression...) from a sample of (possibly enriched) sequences known to belong (or not) to this language (de la Higuera, 2010). This domain is often not very well known due to its roots in theoretical computer science and formal language theory. GI algorithms' known drawback is their lack of efficiency on real data: they are often time consuming, sensitive to errors and do not behave well with large alphabets (for example alphabets containing every word of a natural language).

In this article, we want to give some GI algorithms a chance to compete with the state of the art of statistical machine learning approaches. The task we deal with for this purpose is *chunking* (Abney, 1991) for French, which can be done with hand-made automata (Antoine et al., 2008; Blanc et al., 2010). To our knowledge, trying to *automatically learn these automata* instead of writing them by hand has never been tested for any language before. On the other hand, chunking can also be treated as an annotation task (cf. shared task of CoNLL2000) and thus been efficiently processed by a statistical machine learning approach . The state of the art in this domain are CRFs (Lafferty et al., 2001; Sha and Pereira, 2003). Chunking thus seems to be the ideal playground on which both approaches can be fairly compared.

But this comparison is not our only purpose. Our intuition is that both approaches are complementary, as they focus on very distinct properties of the dataset. We also provide in this article two distinct ways to combine them according to different purposes. The first one is effectiveness-oriented: it consists in enriching the CRF by automata-based features to improve again its effectiveness. The second strategy is readability-oriented: it consists in analyzing the behavior of an automaton produced by GI thanks to CRF-computed weights which are interpretable with respect to this automaton.

The paper is organized as follows. In the first section, we introduce the task of chunking and describe the dataset we have used in all our experi-

Proceedings of Recent Advances in Natural Language Processing, pages 649–658,
Hissar, Bulgaria, 7-13 September 2013.

ments. The second section is dedicated to grammatical inference. After a brief review, we focus on the k-RI-algorithms (Angluin, 1982) and provide the best experimental results we could reach with them on the task. In the next section, we apply CRFs (Lafferty et al., 2001) to the same task. As expected, CRFs give far better results than those obtained by GI, at the price of less readability. In the last section, we describe and evaluate two ways to combine automata and CRFs. The results of both combinations are promising and suggest original trails to associate symbolic and statistical learning.

2 Chunking: the Task and the Data

In this section, we describe the task of chunking as a labeling one and introduce the dataset we used for our experiments. As our purpose is to build a chunker for French, our starting point is the French Treebank (Abeillé et al., 2003).

2.1 The Task

The task of *chunking*, also called *shallow parsing* consists in identifying elementary (i.e. non recursive) syntactic phrases. Chunks are "contiguous and non-recursive lexical units sequences bound to an unique head" (Abney, 1991). Each chunk is characterized by the type (or syntactic category) of its unique head. So, there are as many different types of chunks as there are of considered heads. The chunks are thus intimately linked with the part-of-speech (POS in the following) tags associated with the lexical units of the sentences.

Chunking has been the target of the CoNLL shared task in 2000[1], in which the training set was composed of about 9 000 English sentences taken from the Penn Treebank with two levels of labels: a POS level provided by the Brill tagger and a chunk level. The winners used SVM and "Weighted Probability Distribution Voting". The same corpus was used to show the effectiveness of CRFs (Sha and Pereira, 2003).

2.2 The Data

The French Treebank (FT in the following) has been built from a collection of sentences extracted from articles of the French newspaper "Le Monde", published between 1989 and 1993 (Abeillé et al., 2003). The sentences are tokenized (with respect to some multi-word units), lemmatized, tagged and parsed. There exists multiple versions of the FT, the one we have used is made of about 8 600 XML trees, enriched by syntactic functions which were necessary to identify some chunks. For POS tags, we used the set of 30 morpho-syntactic tags defined by Crabbé and Candito (2008).

We consider 7 distinct types of chunks: AP (adjectival phrases), AdP (Adverbial phrases), CONJ, NP (noun phrases), PP (prepositional phrase), VP (verbal phrases) and UNKNOWN chunks (usually for those containing foreign words). Punctuation marks between chunks are considered as "out". Unlike Tellier et al. (2012), our CONJ chunk only contains the conjunction token(s) and, as opposed to Paroubek et al. (2006), the epithetic adjectives are always part of the NP containing the noun they qualify, whether they appear before or after this noun. Our AP chunk is thus relatively rare, as it only concerns detached or attribute adjectives (syntactic functions available in the XML trees are needed to identify some of them).

An example chunked sentence in our sense is shown in the following (it means "the depreciation against the dollar has been limited to 2.5%")[2][3]:
(la/DET dépréciation/NC)$_{NP}$ (par_rapport_au/P dollar/NC)$_{PP}$ (a/V été/VPP limitée/VPP)$_{VP}$ (à/P 2,5/DET %/NC)$_{PP}$

We extracted from the FT two distinct corpora:

• a corpus where every distinct chunk is extracted and labeled with the BIO (Begin/In/Out) convention. Chunks are distributed according to the following proportions: PP: 33,86%, AdP: 7,23%, VP: 17,11%, AP: 2,21%, NP: 32,95%, CONJ: 6,61%, UNKNOWN: 0,03%.

• a corpus where only NPs are labeled, every other token being considered as out (label O). Recognizing NPs only can be useful for the identification of co-reference chains. This corpus is not a subpart of the previous one, as many PPs include an NP. These "hidden NPs" become visible in the second corpus only, as in the previous example:
(la/DET dépréciation/NC)$_{NP}$ par_rapport_au/P (dollar/NC)$_{NP}$ a/V été/VPP limitée/VPP à/P (2,5/DET %/NC)$_{NP}$

[1] http://www.cnts.ua.be/conll2000/chunking

[2] In this example, NC is the French acronym for CN (common nouns) and VPP is for past participle verbs

[3] A Web page with every detail about the POS and chunk labels (illustrated by many examples) is available but we omit its address here to keep authors anonymous

3 Grammatical inference

Grammatical Inference (GI) is a domain of research which emerged in the 60s and thus has a long history which cannot be easily summed-up. We focus in this section on *GI of automata from positive examples*. After a brief review, we describe the k-RI algorithms (Angluin, 1982) that we used in our experiments and the results we could reach with them.

3.1 Brief state of the art

GI is the study of how it is possible to automatically learn a symbolic device able to represent a *language* (a formal grammar, an automaton...) from a sample of (possibly enriched) sequences known to belong (or not) to this language (de la Higuera, 2010). When only sequences belonging to the language are available, the problem is known as *GI from positive examples*. This is the case in our context, where no counter-example of any kind is available. This problem is much harder than when negative examples are available, because it is very difficult to avoid *over-generalization*. Ultimately, if a learning program hypothesizes that the language to be learned is the universal one (Σ^*, where Σ is the alphabet of the language), no positive example can disprove it, even if it over-generalizes.

The first concern of GI was to provide a precise definition of what it means for a program to be able to "learn a language". The criterion is theoretical and formal, not empirical. A parallel can be drawn with children's language acquisiton. A child is not "programmed" to learn any specific language, (s)he is able to learn whatever language is spoken in his(her) environment. Similarly, GI programs are required to learn *classes of languages*, that is to be able to characterize any member of such a class, when they are provided with examples known to be generated (or not) by this member. The main important "learnability criteria" (also called learning models) are known as "identification in the limit" (Gold, 1967) and "PAC learning" (Valiant, 1984). but we cannot describe them here.

Unfortunately, even for regular languages, the simplest class of the Chomsky hierarchy, those criteria are impossible to fulfill with positive examples only: there is no algorithm able to learn from positive examples the whole class of regular languages satisfying these criteria (Gold, 1967; Kearns and Vazirani, 1994). Researchers have thus tried to identify learnable smaller or transverse classes in Chomsky's hierarchy (Angluin, 1980). k-reversible languages (Angluin, 1982) are such classes, and were the starting point of our experiments. Many other learnable subclasses have been described and studied, for example in Garcia and Vidal (1990; Denis et al. (2002; Kanazawa (1998; Koshiba et al. (2000; Yokomori (2003).

Other advances in the domain concern the learnability of devices integrating probabilities, such as probabilistic automata and their links with HMMs (Thollard et al., 2000; Dupont et al., 2005). In parallel, challenges[4] allowed to test the effectiveness of the proposed algorithms when confronted with real data.

3.2 k-RI Algorithm

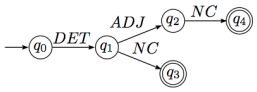
In this section, we describe the GI algorithms used for our experiments. They were applied to try and learn a chunk-specific automaton from the positive sequences of POS tags extracted from the training part of the dataset. GI algorithms from positive examples seem adapted to this problem, as the considered alphabet is limited (30 distinct tags at most) and each distinct kind of chunk can be described by a relatively limited number of syntactic constructions.

k-Reversible Inference (k-RI) algorithm (Angluin, 1982) has the property of identifying in the limit any k-reversible language, for any fixed $k \in \mathbb{N}$. The class of k-reversible languages is a subclass of regular languages, and its members can thus be represented by Deterministic Finite State Automata (DFA). An automaton is k-reversible if it is deterministic and its mirror [5] is *deterministic with a look-ahead of k*. When $k = 0$, a 0-reversible language can be represented by a DFA whose mirror is also deterministic, the algorithm being called Zero Reversible (ZR). If $k_1 < k_2$, the class of k_1-reversible languages is stricty included in the one of k_2-reversible lanugages.

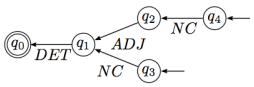
Given a set of positive sequences S, the first step of k-RI is to build PTA(S), the Prefix Tree Acceptor of S. PTA(S) is a tree-shaped DFA, and it has the property of being the smallest tree-shaped DFA recognizing exactly the language defined by S. The root of PTA(S) is its initial state. The search

[4] The most recent ones were Stamina (http://stamina.chefbe.net) and Zulu (http://labh-curien.univ-st-etienne.fr/zulu)

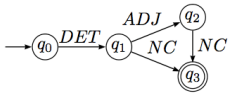
[5] The mirror automaton is obtained by switching initial and final states and by reversing every transition



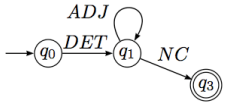
Step 1 : PTA(S)



Step 1 : mirror of PTA(S)



Step 2 : final states of PTA(S) are merged



Step 3 : q_1 and q_2 are merged

Figure 1: Step by step demo of ZR

space of a GI algorithm for a given training set S of positive examples is a lattice whose bottom element is PTA(S) and top element is the universal language built from the alphabet of the examples (Dupont et al., 1994). Most GI algorithms start by building the PTA of the set of available positive examples, then try to generalize the recognized language by merging some of the states of this automaton. k-RI, detailled below, works accordingly. The merging operation here is deterministic, as it propagates recursively through the automaton to preserve determinism.

Algorithm k-RI
In : S : a set of (positive) sequences, k : natural;
Out : A : a k-reversible automaton;
begin
A := PTA(S);
while not(A k-reversible) **do**
// let N1 and N2 be two nodes
// violating k-reversibility of A.
Deterministic_Merge(A, N1, N2);
end while;
return A;
end k-RI;

In Figure 1, we illustrate how ZR behaves with the following set of positive examples of sequences of POS tags: $S = \{DET\ NC,\ DET\ ADJ\ NC\}$. On this example, we see that ZR already generalizes PTA(S) to output an automaton recognizing the language defined by the regular expression: $DET\ ADJ^*\ NC$. This generalization is linguistically relevant. But if we add to the previous positive sample the sequence made of NC alone, ZR will output an automaton recognizing the language $\{DET|ADJ\}^*NC$, which is a more doubtful generalization.

3.3 GI Experience Results on NP Chunking

We applied k-RI for different values of k ($k = 0, k = 1, k = 2$) on POS tags sequences matching NP chunks in the corpus of NP chunks only. This task is the one for which GI is the most appropriate. It is also possible to learn chunk-specific automata on the other corpus, but the application of multiple automata on new data pose a frontier covering problem. Therefore, we only use them in combination with a statistical model, in section 5.

ZR is really sensitive to the available data. A single incorrect sequence can force many states to merge. It was often the case with our dataset, where outliers or tagging errors are not absent. But some erroneous examples can be easily detected: for example, sequences of tags for a special kind on chunk which do not even contain any possible head tag of this chunk can be removed. Other cleaning strategies have been tried. Removing any sequence that occurs less than a fixed proportion was the most effective. Some information loss was nevertheless inevitable as there are heads that were rare (some clitics, for example).

Our experiments were made following a 5-fold cross-validation protocol. A learnt automaton is used as a regular expression on every new sequence of POS tags, looking for the smallest (resp. longest) matches (sm resp. lm). The correctness of a chunk is evaluated in a strict sense, i.e. it is correct if and only if both frontiers are correct. The precision, recall and F1-measure of NP-chunks are computed without taking into account O labels. Table 1 contains various F-measures that we managed to obtain by GI only on NP chunking, with a longest match strategy. Cleaned (c) versions are obtained by deleting every POS sequence that appeared strictly less than 0.01%. Values between parentheses are the medium sizes (in numbers of states) of the 5 automata sizes. PTA versions, whose performances are sometimes good, can be seen as "learning by heart" devices, as they are not generalized. Automata of size 1 are those, probably overgeneralized, that recognize the universal language of POS tags present at least once in NP chunks. $k \geq 2$ is necessary to obtain an automaton behaving better than the cleaned PTA.

4 Statistical learning for annotation

In this section, we focus on the best up-to-date statistical approach to perform an annotation task: Conditional Random Fields (or CRFs). We also

xp	pure PTA	cleaned PTA
F1-meas.	51.92	88.05

xp	c 0-RI (1)	c 1-RI (19)	c 2-RI (68.6)
F1	26.95	72.74	88.25

Table 1: GI results for NP chunking

recall how some HMMs can be "transformed" into a CRF, as it will be useful further.

4.1 Conditional Random Fields and HMMs

CRFs have been introduced by Lafferty et al. (2001). They belong to the family of graphical models. When the graph is linear (which is most often the case), the probability distribution that the annotation sequence y is associated with the input sequence x is expressed by:

$$p(y|x) = \frac{1}{Z(x)} \prod_t \exp\Big(\sum_{k=1}^{K} \lambda_k f_k(t, y_t, y_{t-1}, x)\Big)$$

Where $Z(x)$ is a normalization factor depending on x. This computation is based on K features f_k (usually binary functions), provided by the user. The feature f_k is activated (i.e. $f_k(t, y_t, y_{t-1}, x) = 1$) if a configuration occurring at the current position t in the sequence, concerning y_t, y_{t-1} (i. e. the values of the annotation at the positions t and $t-1$) and x is observed. Each feature f_k is associated with a weight λ_k which are the parameters of the model, to be estimated during the learning step. To define large enough a set of features, softwares implementing CRFs help users: they usually only require to provide *feature templates* which are automatically instanciated into as many features as there are positions in the training data where they can apply. The most current efficient implementation of linear CRFs is Wapiti[6], which uses a L1 penalization allowing to select the best features during the learning step (Lavergne et al., 2010). It is the software we have used.

CRFs have been applied with great success to various annotation tasks, among which POS labeling (Lafferty et al., 2001), named entity recognition (McCallum and Li, 2003), chunking (Sha and Pereira, 2003) and even full parsing (Finkel et al., 2008; Tsuruoka et al., 2009). Their main drawback is that they appear as "black boxes". A CRF model is simply characterized by a list of weighted features but it is not unusual that it contains thousands, even millions of such features. The result is therefore not easy to interpret.

HMMs, which were the previous state of the art for annotation tasks, have the merit to be more understandable. However, every discrete HMM can be "transformed" into a CRF model defining exactly the same probability distribution (Sutton and McCallum, 2006; Tellier and Tommasi, 2011). To do this, you have to define two families of features:

- features of the form $f(y_t, x_t)$ associating an individual label y_t with an individual input x_t: they correspond to the states y_t of the HMM where x_t can be emitted;
- features of the form $f(y_{t-1}, y_t)$ corresponding to the transitions of the HMM linking the states y_{t-1} and y_t.

If θ is a probability of emission or of transition of the HMM, then choose $\lambda = log(\theta)$ as the weight of the corresponding feature in the CRF. The computation of $p(y|x)$ then writes exactly the same in both cases. Discrete HMMs can thus been seen as a special case of CRFs. But CRFs are more general because they allow features to be more general than those used in this transformation. This transformation inspired us to use CRFs to *analyse* a discrete automaton learned by GI. This will be studied in section 5. Before, we provide the learning results obtained by using a CRF on our data.

4.2 Experimental Results

Tables 2 shows the feature templates and results obtained by using CRFs alone on both chunking tasks. For these experiments, we also followed a 5-fold cross-validation protocol and evaluated the chunks in a strict sense. For the complete chunking task, we computed both the micro-average of F-measures (i.e. the average of the F-measures of every kind of chunk weighted by their frequencies) and their macro-average (i.e. without any weight). As expected, CRFs provide excellent results. It is to be noted that they use words in their features along with POS tags, while GI algorithms have only access to the latter.

5 Combinations

In the previous sections, we have applied either pure symbolic learning or pure statistical learning. As expected, symbolic learning provides readable

[6] http://wapiti.limsi.fr/

Feat	Type	Window
Word	Unigram	[-2..1]
POS	Bigram	[-2..1]

chunking	Complete	NP only
micro	97.53	N/A
macro	90.49	N/A
F1-measure	N/A	96.43

Table 2: Template and obtained results with CRFs for each task

word	POS	NP	VP	PP	...	correct label
la	DET	B	O	O	...	B-NP
dépréciation	NC	I	O	O	...	I-NP
par_rapport_au	P	O	O	B	...	B-PP
dollar	NC	B	O	I	...	I-PP
a	V	O	B	O	...	B-VP
été	VPP	O	I	O	...	I-VP
limitée	VPP	O	I	O	...	I-VP
à	P	O	O	B	...	B-PP
2,5	DET	B	O	I	...	I-PP
%	NC	I	O	I	...	I-PP

Table 3: Dataset Enriched by the Output of the chunk-specific Automata

but not very effective programs, whereas it is the contrary for statistical learning. In this section, we want to combine both strategies. There are two different possible viewpoints for this combination:

• if we stand from the viewpoint of effectiveness, we will favor statistical leaning. But the automata provided by our GI algorithms capture long-distance relationships between POS tags that could be useful for a CRF. So, in this case, our combination strategy will consist in integrating the output provided by the automata into the features of the CRF as an external resource.

• if we stand from the viewpoint of readability, we will favor the automata produced by GI. As evoked in 4.1, it is possible to simulate a HMM (and, similarly, an automaton) with CRF's features. We will show that it is also possible to evaluate the states and transitions of an automaton with CRF-computed weights associated to the features that represent them in a CRF, suggesting ways to improve it.

5.1 Enriching a CRF by automata-based features

We attack here both types of chunking. The first combination consists in considering the automata as independent annotation tools, as in Constant and Tellier (2012). In the case of complete chunking, we applied GI on each distinct type of chunk, leading to as many automata as there are types of chunks. Each chunk-specific automaton provides an independent BIO tagging, as shown in Table 3. Therefore, there are as many new attributes as there are types of chunks in our data.

First tables in Tables 4 and 5 give the templates used to obtain the best results for the complete chunking, and similarly for the first one of Table 6 for the NP-chunks only. The lines "Automaton" take into account the output of each automaton independently, whereas "POS+Automata" represents the concatenation of POS columns along with the output of every single automaton.

Matching results are given in the other tables. They show that attributes taken from automata allow to significantly improve the results of CRFs. It is even more obvious for the macro-average, the one that gives equal importance to every chunk. This means that the information brought by the automata mostly improve the recognition of rare chunks. In the experiment leading to the best macro-average, the best improvements are the following: the F1-measure of UNKNOWN goes from 41.67 to 61.22, the one of AP from 96.78 to 97.44 and the one of AdP from 98.72 to 98.92.

Feature	Type	Window
Word	Unigram	[-2..1]
POS	Bigram	[-2..1]
Automaton	Bigram	[-2..1]

F-measure	pure 1-RI (lm)
micro	97.66
macro	92.22

Table 4: Best micro-aver. for complete chunking

Feature	Type	Window
Word	Unigram	[-2..1]
POS	Bigram	[-2..1]
Automaton	Unigram	[-1..1]
POS+Automata	Bigram	[-1..1]

F-measure	pure 1-RI (sm)
micro	97.62
macro	93.52

Table 5: Best macro-aver. for complete chunking

Feature	Type	Window
Word	Unigram	[-2..1]
POS	Bigram	[-2..1]
Automaton	Bigram	[-1..1]
POS+Automata	Bigram	[-1..1]

	pure 2-RI LM
F-measure	96.75

Table 6: Best F-measure for NP chunking

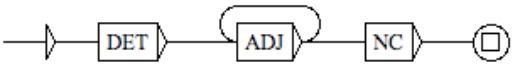


Figure 2: Unitex-generated automaton

5.2 Evaluating an automaton by CRF-computed weights

This time, we want to preserve the structure of the automata output by our GI strategies, but we use a CRF to evaluate some of their properties. We could build weighted automata, the way it is proposed by Roark and Saraclar (2004). Instead, we just propose a CRF-based diagnosis of a purely symbolic device. To illustrate our approach, we consider the NP-only chunking task, because only one automaton is to be considered. Our proposition is also easier to understand by representing automata in the alternative way of Figure 2 (representing the same automaton as the final one of Figure 1). This representation, which is favored in softwares like Unitex[7], has the advantage of displaying tags and transitions between tags as two distinct objects. To build a CRF based on such an automaton, we consider the BIO labeling effect of this automaton, as in section 5.1

Now, inspired by the relationship between discrete HMMs and CRFs (cf. section 4.1), we choose features which can be interpretable relatively to the automaton. We thus restrict ourselves to only two feature-templates:

• the unary feature-template only takes into account the current correct BIO NP-label together with the current POS tag and the current BIO label predicted by the automaton at the same position. Each POS tag matches one (or multiple) states of the automaton. If both BIO labels match for a given POS tag, then the features generated by this template express the correctness of the automaton at this position; if they are different they express its incorrectness

• the bigram feature-template only takes into account the current correct couple of BIO NP-labels together with the corresponding couple of consecutive POS tags as well as the corresponding couple of BIO automaton-predicted labels. The couples of consecutive POS tags characterize transitions of the automata. If the corresponding two couples of BIO labels coincide, it means that the automaton has correctly treated this transition, otherwise it has not.

Note that words, which do not appear in automata, are neither not taken into account in the feature-templates. The generated features have a constrained form to match the automaton structure. All of them are interpretable with respect to this automaton, as we will see now.

Table 7 is a confusion matrix comparing "automata-generated" BIO labels (AL) with the corresponding correct BIO label (CL), for a given POS tag. We can build as many such tables as there are distinct POS tags in NP chunks (the DET tag, in our example), each cell corresponding to an unigram feature. The cells of 7 are filled with the weights computed by the CRF for these features, where the colors display how they can be interpreted with respect to the initial automaton. As expected, weights on the diagonal, meaning a correct tagging, are positive and greater than those outside it, meaning a tagging error.

AL \CL	B	I	O
B	1.66	*-4.05*	*-0.84*
I	**-0.44**	0.46	*-2.51*
O	**-1.45**	**-1.02**	-0.17

Table 7: Confusion matrix for DET tag (2-RI, Table 1)

Where each cell can be interpreted as follows:
• no style : both outputs are identical.
• *italic* : premature chunk beginning.
• **bold** : missed chunk beginning.
• *italic* : untimely chunk continuation.
• **bold** : premature chunk ending.

Bigram features are a bit more complicated to interpret, but they can also give rise to confusion matrices. There are as many bigram confusion matrices as there are observed transitions between two tags, i.e. as many as observed couples of consecutive POS tags (at most $30 * 30$ in our case). A bigram confusion matrix for a specific transi-

[7] http://www-igm.univ-mlv.fr/ unitex/

Exp.	baseline (GI)	0-RI	1-RI	2-RI
chunk	88.25	93.00	93.07	93.08

Table 8: Labeling results of the CRFs based on the best automata for NP-chunking

tion has 9 lines and 9 columns, because there are $9 = 3 * 3$ distinct possible couples of BIO labels. Each cell corresponds to a bigram feature and is interpretable with respect to the transitions of the NP automaton. Each cell can thus also be filled with the weights associated to the corresponding feature by the CRF model.

The weights associated to the features in a CRF characterize their *discriminative power*. They are more relevant than the simple occurrence counts of how many times the features are satisfied in the training dataset. The content of diagonal cells can thus be seen as a measure of the effectiveness of the decision taken by the automaton at a state (resp. a transition) whereas the content of the other cells can be seen as the gain (or loss) taken by using an alternative decision at any time, during the labeling process. So, the whole set of confusion matrixes can be seen as a very precise evaluation of the relevance of the automaton.

Table 8 recalls the result of the best "pure GI" NP-automaton of section 3.3 and gives the labeling result of the CRFs defined as described above on the best automata output by k-RI, for each value of k. We see that the CRFs significantly improve the efficiency of the best automata, but are not as effective as a CRF using more attributes and features. This results can be interpreted as follows: it is sometimes beneficial to take labeling decisions which are not those of the automata. We still haven't taken the time to analyze the various confusion matrices produced by our CRFs in these cases, but we believe that they give very interesting indications about how, where and why the automata on which the features are based made right vs. wrong predictions, and possibly correct them.

6 Conclusion and perspectives

In this paper, we have applied two distinct machine learning approaches on the same dataset and proposed two distinct ways to combine them.

About GI alone, it is possible that other algorithms would give better results than k-RI, such as those of Garcia and Vidal (1990; Denis et al. (2002). The choice of a greater value of k could also improve our results, but at the cost of a greater time complexity[8]. More generally, it should be necessary be to define a learnable language class to which chunks are likely to belong. This would allow to define specific GI algorithms for this task, in which for example linguistic knowledge could be used to "control" state merges .

But the most original part of our work concerns CRFs and automata combinations. It is to be noted that they can both be applied to hand-made automata, likely to be more linguistically relevant than those obtained by GI. We focused here on automata produced by machine learning to show that, even without any linguistic expertise, it is possible to combine symbolic and statistical models. The intuition behind this work is that both machine learning techniques have complementary properties and should benefit from one another. CRFs are based on a huge number of weighted local configurations. It is theoretically possible to express in their features complex long-distance properties of the initial sequence x. In practice, it is rarely done. GI on the contrary applies to sequences and is able to provide a generalization of a set of sequences. It has already been observed that CRFs benefit from features expressing more general properties than simple local configurations (Pu et al., 2010). Our intuition was that GI could provide such useful generalizations. The obtained results confirm this intuition. It is also interesting to see that symbolic models enhance the treatment of rare cases, on which statistical models do not behave well.

CRF-generated confusion matrices for the analysis of an automaton still need to be further investigated. How to better interpret or take advantage of them is of particular interest. Some of the cells of these matrices are empty, either because the corresponding feature has not been observed in the training set or because it has been discarded by Wapiti during the learning step because of the penalty. It should be possible, thanks to this information, to modify the automaton on which the CRF is based by removing/adding states or transitions according to the diagnosis of the confusion matrices. A CRF-directed GI strategy still needs to be defined. This kind of GI challenge could also benefit from existing learning algorithms targeting probabilistic automata (Thollard et al., 2000).

[8] k-RI time complexity is of $|\Sigma|^k|Q|^{k+3}$ where $|Q|$ is the number of states of the PTA.

References

[Abeillé et al.2003] A. Abeillé, L. Clément, and F. Toussenel. 2003. Building a treebank for french. In A. Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

[Abney1991] S Abney. 1991. Parsing by chunks. In R. Berwick, R. Abney, and C. Tenny, editors, *Principle-based Parsing*. Kluwer Academic Publisher.

[Angluin1980] D. Angluin. 1980. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, May.

[Angluin1982] D. Angluin. 1982. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, July.

[Antoine et al.2008] Jean-Yves Antoine, Abdenour Mokrane, and Nathalie Friburger. 2008. Automatic rich annotation of large corpus of conversational transcribed speech: the chunking task of the epac project. In *Proceedings of LREC'2008*, may.

[Blanc et al.2010] Olivier Blanc, Matthieu Constant, Anne Dister, and Patrick Watrin. 2010. Partial parsing of spontaneous spoken french. In *Proceedings of LREC'2010*.

[Constant and Tellier2012] M. Constant and I. Tellier. 2012. Evaluating the impact of external lexical resources unto a crf-based multiword segmenter and part-of-speech tagger. In *Proceedings of LREC 2012*.

[Crabbé and Candito2008] B. Crabbé and M. H. Candito. 2008. Expériences d'analyse syntaxique statistique du franÃ§ais. In *Actes de TALN'08*.

[de la Higuera2010] C. de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

[Denis et al.2002] F. Denis, A. Lemay, and A. Terlutte. 2002. Some language classes identifiable in the limit from positive data. In *ICGI 2002*, number 2484 in Lecture Notes in Artificial Intelligence, pages 63–76. Springer Verlag.

[Dupont et al.1994] P. Dupont, L. Miclet, and E. Vidal. 1994. What is the search space of the regular inference. In Lecture Notes in Artificial Intelligence, editor, *ICGI'94 - Lectures Notes in Computer Science*, volume 862 - Grammatical Inference and Applications, pages 25–37, Heidelberg.

[Dupont et al.2005] Pierre Dupont, François Denis, and Yann Esposito. 2005. Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349–1371.

[Finkel et al.2008] Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008)*, pages 959–967.

[Garcia and Vidal1990] P. Garcia and E. Vidal. 1990. Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):920–925.

[Gold1967] E.M. Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

[Kanazawa1998] M. Kanazawa. 1998. *Learnable Classes of Categorial Grammars*. The European Association for Logic, Language and Information. CLSI Publications.

[Kearns and Vazirani1994] M. J. Kearns and U. V. Vazirani. 1994. *An Introduction to Computational Learning Theory*. MIT Press.

[Koshiba et al.2000] Takeshi Koshiba, Erkki Mäkinen, and Yuji Takada. 2000. Inferring pure context-free languages from positive data. *Acta Cybernetica*, 14(3):469–477.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

[Lavergne et al.2010] Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of ACL'2010*, pages 504–513. Association for Computational Linguistics, July.

[McCallum and Li2003] A. McCallum and W. Li. 2003. Early results for named entity recognition with conditional random fields. In *CoNLL'2003: Proceedings of The Seventh Conference on Natural Language Learning*.

[Paroubek et al.2006] P. Paroubek, I. Robba, A. Vilnat, and Ayache C. 2006. Data annotations and measures in easy, the evaluation campain for parsers of french. In *Proceedings of LREC'2006*, pages 315–320.

[Pu et al.2010] X. Pu, Q. Mao, G. Wu, and C. Yuan. 2010. Chineese named entity recognition with the improved smoothed conditional random fields. *Research in Computing Science*, 46:90–103. Special issue "Natural Language Processing and its Applications".

[Roark and Saraclar2004] Brian Roark and Murat Saraclar. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL*, pages 47–54.

[Sha and Pereira2003] F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*, pages 213 – 220.

[Sutton and McCallum2006] Charles Sutton and Andrew McCallum, 2006. *Introduction to Statistical Relational Learning*, chapter An Introduction to Conditional Random Fields for Relational Learning. MIT Press, lise getoor and ben taskar edition.

[Tellier and Tommasi2011] Isabelle Tellier and Marc Tommasi. 2011. Champs Markoviens Conditionnels pour l'extraction d'information. In Eric Gaussier and François Yvon, editors, *Modèles probabilistes pour l'accès à l'information textuelle*. Hermès.

[Tellier et al.2012] I. Tellier, D. Duchier, I. Eshkol, A. Courmet, and M. Martinet. 2012. Apprentissage automatique d'un chunker pour le français. In *Actes de TALN'12, papier court (poster)*.

[Thollard et al.2000] Franck Thollard, Pierre Dupont, and Colin de la Higuera. 2000. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proc. 17th International Conf. on Machine Learning*, pages 975–982. Morgan Kaufmann.

[Tsuruoka et al.2009] Y. Tsuruoka, J. Tsujii, and S. Ananiadou. 2009. Fast full parsing by linear-chain conditional random fields. In *Proceedings of EACL 2009*, pages 790–798.

[Valiant1984] L.G. Valiant. 1984. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November.

[Yokomori2003] T. Yokomori. 2003. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 1.