Even better than Supertags : Introducing Hypertags !

Alexandra Kinyon

TALaNa – LaTTice
UFRL, University Paris 7, case 7003
2, pl. Jussieu
F-75251 Paris Cedex 05
Email : Alexandra.Kinyon@linguist.jussieu.fr

Abstract

In this paper, we introduce the notion of Hypertag, which allows to factor the information contained in several Supertags into a single structure. We also discuss why this approach is useful within frameworks other than LTAGs, and how it can be used for annotating and searching corpora.

Introduction

Traditional part of speech tagging assigns very limited information (i.e. morphological and local) to lexical items, thus providing only limited help for parsing. To solve this problem, (Joshi & Srinivas 94, Srinivas 97) extend the notion of POS by introducing Supertags, within the framework of Lexicalized Tree Adjoining Grammars (LTAGs). Unfortunately, words are assigned on average a much higher number of Supertags than traditional POS : On average for English a word is associated with 1.5 POS and with 9 supertags (Joshi 99). One common solution to the problem is to only retain the "best" supertag for each word, or eventually the 3 best supertags for each word, which is what (Srinivas 97) does in a probabilistic manner. But then, early decision has an adverse effect on the quality of parsing if the wrong supertag(s) have been kept : one typically obtains between 75% and 92% accuracy when keeping only one supertag / item (depending on the type of text being supertagged and on the technique used) (cf. Srinivas 97, Chen & al. 99) which means that it may be the case that every word in 4 will have a wrong supertag, whereas typical POS taggers usually achieve an accuracy above 95%.

Solutions for packing several supertags into a single structure have been proposed in the past, for example by resorting to logical formulae (Kallmeyer 99) or linear types of trees (Halber 99). But as argued in (Kinyon 00a), these solutions are unsatisfactory because they rely only on mathematical properties of trees, and lack a linguistic dimension.

In this paper, we introduce the notion of Hypertag, which allows to factor the information contained in several Supertags, so that a single structure can be assigned to each word. In addition of being well-defined computational objects, hypertags should also be "readable" and also motivated from a linguistic point of view . In a first part, we explain the solution we have adopted, building up on the notion of MetaGrammar introduced by (Candito 96) & (Candito, 99). Finally, we discuss how this approach can be used in practice, and why it is interesting for frameworks other than LTAGs. We assume the reader is familiar with LTAGs and Supertags and refer respectively to (Joshi 87) & to (Srinivas 97) for an introduction.

1. Exploiting a MetaGrammar

(Candito 96,99) has developed a tool to generate semi-automatically elementary trees She uses an additional layer of linguistic description, called the metagrammar (MG), which imposes a general organization for syntactic information in a 3 dimensional hierarchy :

- **Dimension 1:** initial subcategorization

- **Dimension 2:** redistribution of functions and transitivity alternations
- **Dimension 3:** surface realization of arguments, clause type and word order

Each terminal class in dimension 1 describes a possible initial subcategorization (i.e. a tree family). Each terminal class in dimension 2 describes a list of ordered redistributions of functions (e.g. it allows to add an argument for causatives). Finally, each terminal class in dimension 3 represents the surface realization of a (final) function (e.g. cliticized, extracted ...).

Each class in the hierarchy corresponds to the partial description of a tree (cf. Rogers & Vijay-Shanker 94). An elementary tree is generated by inheriting from one terminal class in dimension 1, from one terminal class in dimension 2 and from n terminal classes in dimension 3 (were n is the number of arguments of the elementary tree). [1] The hierarchy is partially handwritten. Then crossing of linguistic phenomena (e.g. passive + extraction), terminal classes and from there elementary trees are generated automatically off line[2]. This allows to obtain a grammar which can then be used to parse in real time. When the grammar is generated, it is straight forward to keep track of the terminal classes each elementary tree inherited from : Figure 1 shows seven elementary trees which can supertag "donne" (gives), as well as the inheritance patterns[3] associated to each of these supertags. All the examples below will refer to this figure.

The key idea then is to represent a set of elementary trees by a disjunction for each dimension of the hierarchy. Therefore, a hypertag consists in three disjunctions (one for dimension 1, one for dimension 2 and one for dimension 3). The cross-product of the three disjunctions can then be done automatically and from there, the set of elementary trees referred to by the hypertag will be automatically retrieved We will now illustrate this, first by showing how hypertags are built, and then by explaining how a set of trees (and thus of supertags) is retrieved from the information contained in a hypertag.

1.1 Building hypertags : a detailed example

Let us start with a simple exemple were we want "donner" to be assigned the supertags $\alpha 1$ (*J. donne une pomme à M.*) and $\alpha 2$ (*J donne à M. une pomme*). On figure 1, one notices that these two trees inherited exactly from the same classes : the relative order of the two complements is left unspecified in the hierarchy, thus one same description will yield both trees. In this case, the hypertag will thus simply be identical to the inheritance pattern of these two trees :

Dimension 1:	n0vn1(àn2)
Dimension 2 :	no redistribution
Dimension 3	subj :nominal-canonical obj : nominal-canonical a-obj: nominal-canonical

Let's now add tree $\alpha 3$ (*J. donne une pomme*) to this hypertag.This tree had its second object declared empty in dimension 2 (thus it inherits only two terminal classes from dimension 3, since it has only two arguments realized). The hypertag now becomes[4] :

---

[1] The idea to use the MG to obtain a compact representation of a set of SuperTags was briefly sketched in (Candito 99) and (Abeillé & al. 99), by resorting to MetaFeatures, but the approach here is slightly different since only information about the classes in the hierarchy is used (and not explicit information about the function of arguments)

[2] This point has been misunderstood by (Xia & al. 98, p.183) : terminal classes and classes for crossings of phenomena ARE NOT manually created

[3] We call inheritance patterns the structure used to store all the terminal classes a tree has inherited from.

[4] What has been added to a supertag is shown in bold characters.

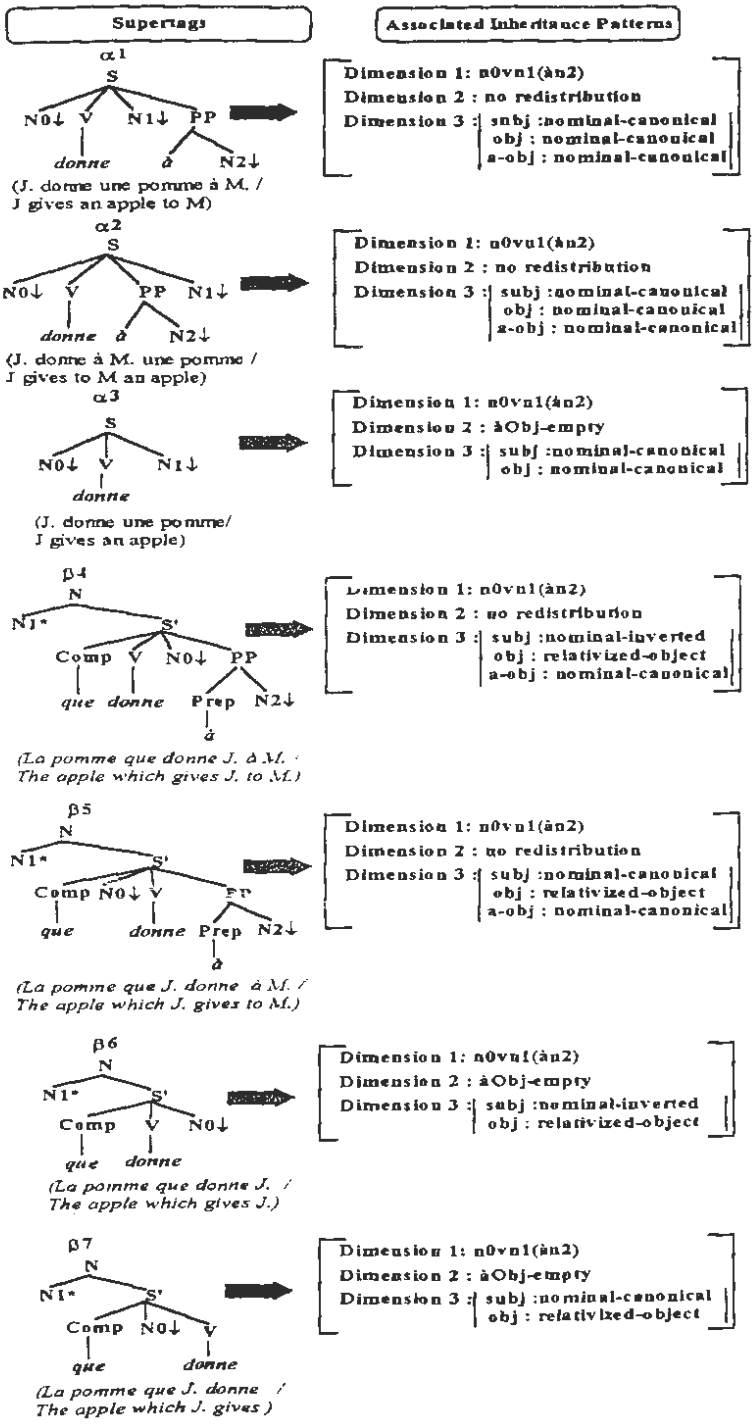

FIGURE 1 : SuperTags and associated inheritance patterns

Dim. 1:	n0vn1(àn2)
Dim. 2 :	no redistribution **OR àObj- empty**
Dim. 3	subj :nominal-canonical obj : nominal-canonical a-obj: nominal-canonical

Let's now add the tree β4 for the object relative to this hypertag. This tree has been generated by inheriting in dimension 3 from the terminal class "nominal inverted" for its subject and from the class "relativized object" for its object. This information is simply added in the hypertag, which now becomes :

Dim. :	n0vn1(àn2)
Dim. 2 :	no redistribution OR àObj- empty
Dim. 3	subj :nominal-canonical **OR nominal-inverted** obj : nominal-canonical **OR relativized-object** a-obj: nominal-canonical

Also note that for this last example the structural properties of β4 were quite different than those of α1, α2 and α3 (for instance, it has a root of category N and not S). But this has little importance since a generalization is made in linguistic terms without explicitly relying on the shape of trees.

It is also clear that hypertags are built in a monotonic fashion : each supertag added to a hypertag just adds information. Also, the process of building hypertags is rather simple. We observe that hypertags allow to label each word with a unique structure[5]. Moreover, hypertags contain rich syntactic information about lexical items (For our example, the word "*donne*"), and also contain functional information (not explicitly available in supertags). They are linguistically motivated, but also yield a readable output. They can be enriched or modified by human annotators or easily fed to a parser or shallow parser.

1.2 Retrieving information from hypertags

Retrieving information from hypertags is pretty straightforward. For example, to recover the set of supertags contained in a hypertag[6], one just needs to perform the crossing between the 3 dimensions of the hypertag, as shown on Figure 2, in order to obtain all inheritance patterns. These inheritance patterns are then matched with the inheritance patterns contained in the grammar (i.e. the right column in Figure 1) to recover all the appropriate supertags. Inheritance patterns which are generated but don't match any existing trees in the grammar are simply discarded[7].

We observe that the 4 supertags α1, α2 and α3 and β4 which we had explicitly added to the hypertag in 2.1 are correctly retrieved. But also, the supertags β5, β6 and β7 are retrieved, which we did not explicitly intend since we never added them to the hypertag. But this is not a problem, since if a word can anchor the 4 first trees, then it will also necessarily anchor the three last ones. In fact, the automatic crossing of disjunctions in the hypertag insures consistency[8].

---

[5] We presented a simple example for sake of clarity, but traditional POS ambiguity is handled in the same way, except that disjunctions are then added in dimension 1 as well.

[6] This is to show that supertags can be retrieved from a hypertag. But it is not indispensable to do so : using hypertags directly is more appealing and will be addressed in future work.

[7] When the full 5000 trees grammar is generated with the MetaGrammar, these same trees are discarded by general linguistic principles such as "canonical nominal objects prevent subject inversion" (cf. Abeillé & al. 00). So Hypertags do not "overgenerate".

[8] Again, for the same reasons the MetaGrammar insures consistency.

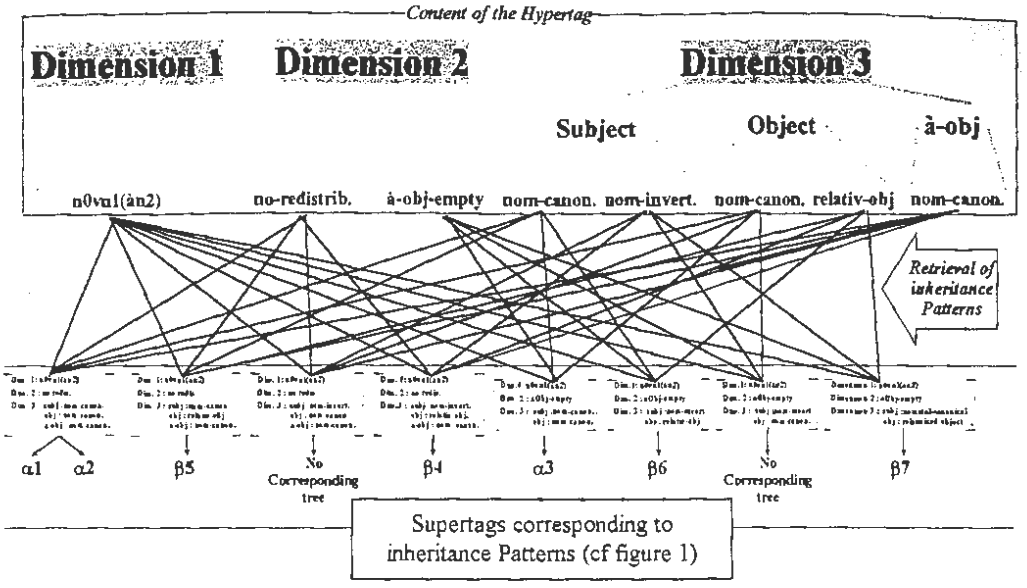

FIGURE 2 :Retrieving inheritance patterns and Supertags from a Hypertag

Also note that no particular mechanism is needed for dimension 3 to handle arguments which are not realized : if àObj-empty is inherited from dimension 2, then only subject and object will inherit from dimension three (since only arguments that are realized inherit from that dimension when the grammar is generated).

Information can also be modified at runtime in a hypertag, depending on the context of lexical items. For example *"relativized_object"* can be supressed in dimension 3 from the hypertag shown on Figure 2, in case no Wh element is encountered in a sentence. Then, the correct set of supertags will still be retrieved from the hypertag by automatic crossing (that is, trees $\alpha 1$,$\alpha 2$ and $\alpha 3$), since the other inheritance patterns generated won't refer to any tree in the grammar (here, no existing tree inherits in dimension 3 *"subject:inverted-nominal"*, without inheriting also *"object: relativized-object"*)

2. Practical use

An LTAG can be seen as a dictionary, in which each lexical entry is associated to a set of elementary trees. But with hypertags, each lexical entry is now paired with one unique structure. Therefore, automatically hypertagging a text is easy (i.e. simple dictionary lookup). The equivalent of finding the "right" supertag for each lexical item in a text (i.e. reducing ambiguity) then consists in dynamically removing information from hypertags (i.e. suppressing elements in disjunctions). We hope this can be achieved by specific rules, which we are currently working on. It is important to note though that the resulting output can easily be manually annotated in order to build a gold-standard corpus : manually removing linguistically relevant pieces from information in a disjunction from a single structure is simpler than dealing with a set of trees. In addition of obvious advantages in terms of display (tree structures, especially when presented in a non graphical way, are unreadable), the task itself becomes easier because topological problems are solved automatically: annotators need just answer questions such as *"does this verb have an extracted object ?"*, *"is the subject of this verb*

inverted ?" to decide which terminal classe(s) must be kept[9] .We believe that these questions are easier to answer than *"Which of these trees have a node N1 marked wh+ at address 1.1 ?"* (for an extracted object).

Also, supertagged text are difficult to use outside of an LTAG framework, contrary to hypertagged texts, which contain general linguistic information. An example would be searching and extracting syntactic data on a large scale : suppose one wants to extract all the occurrences where a given verb V has a relativized object. To do so on a hypertagged text simply involves performing a "grep" on all lines containing a V whose hypertag contains *"dimension 3 : objet:relativized"*, without knowing anything about the LTAG framework. Performing the same task with a supertagged text involves knowing how LTAGs encode relativized objects in elementary trees, scanning potential trees associated with V... Another example would be using a hypertagged text as an input to a parser based on a framework other than LTAGs : for instance, hypertags could be used by an LFG parser to constrain the construction of an F-structure, whereas it's unclear how this could be achieved with supertags.

3. Conclusion

We have introduced the notion of hypertag. Hypertags allow to assign one unique structure to lexical items. Moreover this structure is readable, linguistically and computationally motivated, and contains much richer syntactic information than traditional POS, thus a hypertagger would be a good candidate as the front end of a parser. It allows in practice to build large annotated resources which are useful for extracting syntactic information on a large scale, without being dependant on a given grammatical formalism. Also, hypertags are being used to develop a psycholinguistically motivated processing model for LTAGs (Kinyon 00b).

We have shown how hypertags are built, how information can be retrieved from them. Further work will investigate how hypertags can be combined directly.

References

Abeillé A., Candito M., Kinyon A. 1999. FTAG: current status and parsing scheme. Proc. Vextal '99. Venice.

Abeillé A. Candito M.H., Kinyon A. 2000. The current status of FTAG. Proc. TAG+5. Paris.

Candito M-H. 1996. A principle-based hierarchical representation of LTAGs. Proc. COLING'96 Kopenhagen.

Candito M.-H. 1999. Représentation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien. PhD dissertation. University Paris 7.

Chen J., Srinivas B., Vijay-Shanker K. 1999 New Models for Improving Supertag Disambiguation. Proc. EACL'99 pp. 188-195. Bergen.

Halber A. 1999. Stratégie d'analyse pour la compréhension de la parole : vers une approche à base de Grammaires d'Arbres Adjoints Lexicalisées. PhD thesis. ENST. Paris

Joshi A. : 1987. *An introduction to Tree Adjoining Grammars.* In Mathematics of Language. A. Manaster-Ramer (eds). John Benjamins Publishing Company. Amsterdam.Philadelphia. pp. 87-114.

Joshi A. 1999. Explorations of a domain of locality. CLIN'99. Utrecht.

Joshi A. Srinivas B. 1994. Disambiguation of Super Parts of Speech (or Supertags) : Almost parsing. Proceeding COLING'94. Kyoto.

Kallmeyer L. 1999, Tree Description Grammars and Underspecified Representations. PhD thesis. Universität Tübingen.

Kinyon A. 2000a. Hypertags. Proceedings Coling'00. Sarrebrucken.

Kinyon A. 2000b. Towards a psycholinguistically motivated processing model for LTAGs. Proc. Cogsci' 2000. Philadelphia.

Srinivas B., 1997. Complexity of lexical descriptions and its relevance for partial parsing, PhD thesis. Univ. of Pennsylvania.

Rogers J., Vijay-Shanker K. 1994. Obtaining trees from their descriptions : an application to TAGs. Computational Intelligence. 10:4 pp 401-421.

Xia F. & Palmer M., Vijay-shanker K., Rosenzweig J. 1998. Consistent Grammar development using partial tree description for LTAGs. Proc. TAG+4. Philadelphia.

[9] This of course implies that one must be very careful in choosing evocative names for terminal classes.