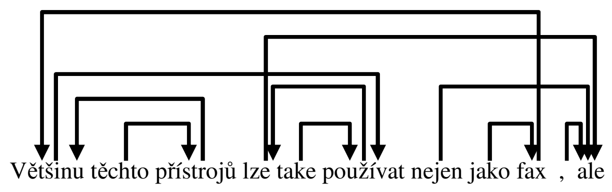
I adopted a novel approach, which consists in adding six new parsing actions:

`Right2`	in a configuration $\langle s_1\|s_2\|S, n\|I, T, A\rangle$, adds an arc from s_2 to n and removes s_2 from the stack, producing the configuration $\langle s_1\|S, n\|I, T, A\cup\{(s_2, r, n)\}\rangle$.
`Left2`	in a configuration $\langle s_1\|s_2\|S, n\|I, T, A\rangle$, adds an arc from n to s_2, pops n from input, pops s_1 from the stack and moves it back to I, producing the configuration $\langle s_2\|S, s_1\|I, T, A\cup\{(n, r, s_2)\}\rangle$.
`Right3`	in a configuration $\langle s_1\|s_2\|s_3\|S, n\|I, T, A\rangle$, adds an arc from s_3 to n and removes s_3 from the stack, producing the configuration $\langle s_1\|s_2\|S, n\|I, T, A\cup\{(s_3, r, n)\}\rangle$.
`Left3`	in a configuration $\langle s_1\|s_2\|s_3\|S, n\|I, T, A\rangle$, adds an arc from n to s_3, pops n from input, pops s_1 from the stack and moves it back to I, producing the configuration $\langle s_2\|s_3\|S, s_1\|I, T, A\cup\{(n, r, s_3)\}\rangle$.
`Extract`	in a configuration $\langle s_1\|s_2\|S, n\|I, T, A\rangle$, move s_2 from the stack to the temporary stack, then `Shift`, producing the configuration $\langle n\|s_1\|S, I, s_2\|T, A\rangle$.
`Insert`	in a configuration $\langle S, I, s_1\|T, A\rangle$, pops s_1 from T and pushes it to the stack, producing the configuration $\langle s_1\|S, I, T, A\rangle$.

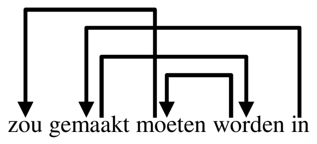
The actions `Right2` and `Left2` are sufficient to handle almost all cases of non-projectivity: for instance the training data for Czech contain 28081 non-projective relations, of which 26346 can be handled by `Left2`/`Right2`, 1683 by `Left3`/`Right3` and just 52 require `Extract`/`Insert`.

Here is an example of non-projectivity that can be handled with `Right2` (*nejen* → *ale*) and `Left3` (*fax* → *Většinu*):

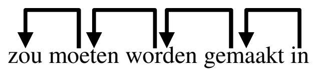
Většinu těchto přístrojů lze take používat nejen jako fax, ale současně …


Většinu těchto přístrojů lze take používat nejen jako fax , ale

The remaining cases are handled with the last two actions: `Extract` is used to postpone the creation of a link, by saving the token in a temporary stack; `Insert` restores the token from the temporary stack and resumes normal processing.



This fragment in Dutch is dealt by performing an `Extract` in configuration ⟨*moeten*|*gemaakt*|*zou*, *worden*|*in*, *A*⟩ followed immediately by an `Insert`, leading to the following configuration, which can be handled by normal `Shift`/`Reduce` actions:



Another linguistic phenomenon is the anticipation of pronouns, like in this Portuguese fragment:

`Tudo é possivel encontrar em o IX Salão de Antiguidades, desde objectos de ouro e prata, moedas, …`

The problem here is due to the pronoun `Tudo` (*Anything*), which is the object of `encontrar` (*find*), but which is also the head of `desde` (*from*) and its preceding comma. In order to be able to properly link `desde` to `Tudo`, it is necessary to postpone its processing; hence it is saved with `Extract` to the temporary stack and put back later in front of the comma with `Insert`. In fact the pair `Extract`/`Insert` behaves like a generalized `Rightn`/`Leftn`, when n is not known. As in the example, except for the case where n=2, it is difficult to predict the value of n, since there can be an arbitrary long sequence of tokens before reaching the position where the link can be inserted.

5 Performance

I used my own C++ implementation of Maximum Entropy, which is very fast both in learning and classification. On a 2.8 MHz Pentium Xeon PC, the learning time is about 15 minutes for Portuguese and 4 hours for Czech. Parsing is also very fast, with an average throughput of 200 sentences per second: Table 1 reports parse time for parsing each whole test set. Using Memory Based Learning increases considerably the parsing time, while as expected learning time is quite shorter. On the other hand MBL achieves an improvement up to 5% in accuracy, as shown in detail in Table 1.