What is Inside Llewyn Davis?*
Inner structure of MWEs

Haifa poster proposal, Working Group 4, work in progress

Eduard Bejček, Pavel Straňák
Charles University in Prague, MFF, ÚFAL
{bejcek,stranak}@ufal.mff.cuni.cz

Prague Dependency Treebank (Bejček et al., 2012) has very comprehensive syntactic annotation. It also includes annotation of spans of multiword expressions (since PDT 2.5,), including multiword named entities (NEs) (Straňák, 2010). What it is still missing is harmonising annotation of these phenomena: answering the question how exactly should MWEs, including their structure, integrate with annotation of syntax.

What we demonstrate in this poster is annotation of structure of selected types of multiword named entities and how this structure relates to the syntactic structures around and inside these MWEs.

We show that some types of NEs, e.g. `address`, contain relatively standard components with rules for their "filling" that can be understood as a grammar. Structure of such NEs can be represented as a table or as a phrase structure. These structures are very regular and such NEs are rather part of the grammar than part of the lexicon. The `address` can embed other NEs in some slots of the table (or nonterminals). For nested named entities see (Ševčíková et al., 2007).

Other types of NEs like `persons` should be often captured in a lexicon and linked to their individual pragmatic attributes (births, deaths, etc.), but `person` also consists of standard components. It is not obvious whether these should be always part of the syntactic description.

Yet other NE types like `locations` do not have this simple set of standard components. At the same time it is important to capture some of their pragmatic attributes. Thus we propose to represent them as atomic nodes with a reference to a lexicon.

Below we give more details on these types of NEs and we illustrate our proposed representation for them.

`location` and `object`

There are place names whose inner structure is no longer important (*"Hong Kong"*), while it should be preserved in others (*"South Carolina"*). Then there are many location names in between (*"Trafalgar Square"*, *"New York"*). `Objects` (names of books, laws, chemicals, units, product names, ...) are a similar case.

We want to keep these named entities in the lexicon and refer to them from the node in the treebank. Besides the advantage of compiled lexicon of named entities itself, the lexicon also provides the way to assign further information to each named entity (e.g. whether it is a river, country or a street; GPS coordinates; link to the Wikipedia web page; etc.) and to describe nested named entities. In Figure 1, it is illustrated using an `object` (a statue) with `personal` and `location` names inside.

`address`

The inner structure of a complex address entry is definitely far from any dependency relation – consider e.g. a telephone num-

* There are relations between *"Llewyn"* and *"Davis"* inside *"Llewyn Davis"* that are not exactly dependency relations; and also the last film by Coen brothers is *Inside Llewyn Davis*, i.e. this entity: `http://en.wikipedia.org/wiki/Inside_Llewyn_Davis`.

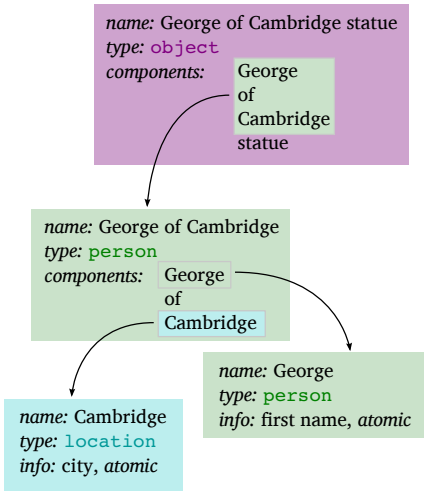

Figure 1: The way to capture nested named entities of an `object` *"George of Cambridge statue"* in a lexicon.

ber or the relation between the institution name and the ZIP code.

We propose representing the whole address as a single node in the dependency tree and to describe its inner structure as a set of attribute-value pairs, i.e. as a two-column table. The table has many (possibly empty) components such as `house number`, `street or square`, `state`, or even `storey` and `e-mail address`. An example of an `address` split into the table is in Figure 2. Moreover, the several parts of an address can refer to another named entities that again are kept separately in the lexicon.

There is a big difference between `address` and `location` or `object`: It is pointless to build a lexicon of `addresses`, since `address` is more a part of a grammar. The filled table (e.g. that from Figure 2) should be directly in the treebank and constitute the `address`-node in the tree.

`person`

We do not see the dependency of the personal name on the family name as a proper relation. It actually does not matter which name is which: if one see *"Zhang Yimou"* (sometimes referred to as *"Yimou Zhang"*), they know it is a name (maybe they even know it is a name of Chinese director), but they do not need to know, which part of

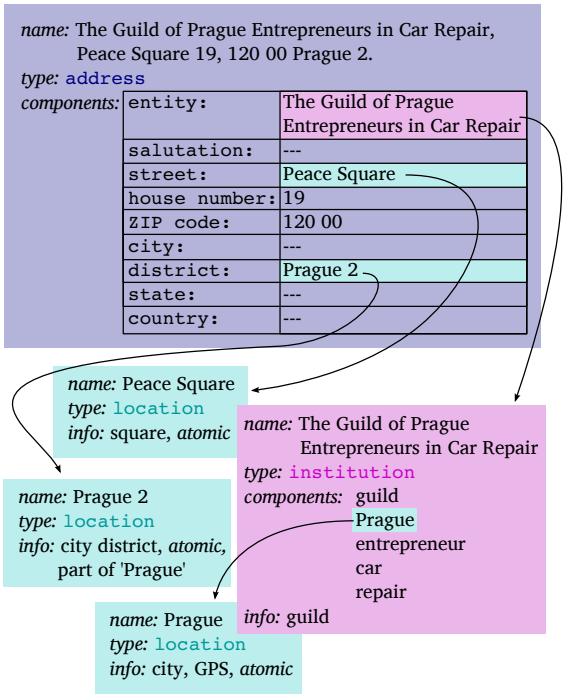

Figure 2: The proposal of an address structure. The dark blue frame is a part of the data; the rest of the figure is a part of the lexicon.

the name is a family name. Similarly to an `address`, the suitable description is a table, but this time the table should be in a lexicon. The components here would be e.g. `family name`, `middle name`, `academic title`, `preposition` (such as *"von"*, *"de"*, *"van"*, ...).

References

Eduard Bejček, Jarmila Panevová, Jan Popelka, Pavel Straňák, Magda Ševčíková, Jan Štěpánek, and Zdeněk Žabokrtský. 2012. Prague dependency treebank 2.5 – a revisited version of PDT 2.0. In Martin Kay and Christian Boitet, editors, *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, pages 231–246, Mumbai, India.

Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza. 2007. Named entities in czech: Annotating data and developing NE tagger. In Václav Matoušek and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 188–195, Berlin / Heidelberg. Springer.

Pavel Straňák. 2010. *Annotation of Multiword Expressions in The Prague Dependency Treebank*. Ph.D. thesis, Charles University in Prague.