Application of Hungarian WordNet in Semantic Category Definition of Nouns for Cluster Analysis

Anonymous Submission

Abstract

In this paper, I would like to present my project regarding the application of Hungarian WordNet (Miháltz et al., 2008) in defining semantic selectional categories for rules Verbs inflict on their nominal Subjects. I will give a proposal and an actual realization of a program that can turn the previously manual preprocessing task into an automated or semi-automated knowledge-based process that can easily account for outlier Subjects or different meanings of Verbs.

1 Introduction

When processing language, we often try to find patterns or groupings that would help explain various phenomena and could bring us closer to understanding the inner workings of natural languages. One of these patterns can be the way different (groups of) Verbs choose their arguments based on some shared aspect of the meanings of said arguments.

The basis of this research is a previous, more linguistically based clustering task, in which I aimed to define verbal categories based on their semantic selectional preferences and their thematic role distribution on the Subject position based on corpus data form the Hungarian Gigaword Corpus (Oravecz et al., 2014) using roughly 70.000 sentences. To keep a smaller scope, this research was limited to nominal Subjects, so Proper Nouns and Pronouns were excluded in this work. I worked with 100 Verbs and 54 Nouns as their possible Subjects; these Nouns were manually categorized into 5 semantic categories – based on my intuition as a native speaker of Hungarian. This last part is where the current research becomes a much-needed addition to the clustering task: although the judgement of a native speaker is fairly reliable, it is not without error and at this low number of used words this was an acceptable way of categorizing, but if the number of used words was increased by just tenfold, it would be a tedious and almost certainly faulty method.

The given categories were also static, which could be an acceptable way if there was a widely accepted complete list of categories of semantical selection. This, however, is not the case. This is why various WordNets can be used for a clearer and better-suited list of categories (Ye, 2004). Moving away from this static understanding of the selectional preferences and towards a knowledge-based solution could be the right move, and using a WordNet is probably the best tool for that goal. Additionally, with the use of WordNet there wouldn't be a need of a limit to a relatively short list of Nouns, rather the scope of the research could be widened to every Subject that occur in the corpus, even taking Proper Nouns into consideration.

The paper has the following structure: following the introduction of the theoretical background in Chapter 2, in Chapter 3 I present the theoretical problem that is faced with this project, with the solution also presented in a schematic manner. In Chapter 4 I expand upon the previous schematic explanation introducing the algorithm that was constructed to tackle this challenge, after which Chapter 5 gives a short overview of the results and limitations of the current version and presents future steps to improve the state of this project.

2 Background

According to Chomsky (1965), in the case of semantic selection we have underlying rules that constrain the fully free choice of arguments, but this rule is not coded in syntax, rather it is part of a further layer, that is outside the domain of grammatical rules, so even though it rules over sentence forming, it is rather the part of semantics. Semantic selection can be formalized based on Resnik (1997), we can understand the selection as the difference between a prior and a posterior distribution, where the prior distribution is the general probability of a word appearing without any restrictions in its environment and the posterior distribution is the conditional probability with the addition of a predicate. If a Verb has weak selectional preference, the difference between the two distributions is small, if it has strong preference, the difference is big.

The basic idea behind this work is that semantical selection can be understood through application of the hypernymy relation. Hypernymy is a basic semantic relation, in which a lexical element with broader meaning is the superordinate of other lexical elements that have narrower meanings that expand the concept of their superordinate, like *fruit* is the hypernym or superordinate of *apple*, *cherry* and *orange* (Bußmann and Trauth, 1996). The relation between words and word classes in the case of hypernymy (both in theory and in WordNet) is understood as an IS-A relation (i.e. *apple* IS A *fruit*) (Resnik, 1993). This relation is transitive, meaning that the hypernym of a hypernym is also the hypernym of the original word that was the subordinate in the first place. Given this nature of the relation, hypernymy can be viewed as a hierarchical tree, where the roots are the most basic aspects of meaning (although being words themselves) and leaves are the end states of the superordination relation, the words with the most specific meanings, with every node in this tree being its own word with its own meaning.

For this project I used the Hungarian WordNet (HuWN) (Miháltz et al., 2008). HuWN has over 40.000 synsets, of which approximately 33.000 are Nouns. HuWN was constructed following the conventions of the original WordNet (Miller, 1995), while adapting the methods to the typological differences that are between English and Hungarian (Vincze et al., 2008). Thus, HuWN is an excellent tool for tasks like this.

3 Proposal

Previously, I had lists of words that needed some kind of tag that helps describe the selectional rules that play a role in choosing those exact words. This tag, as mentioned above, was first part of a static list, a Verb could only choose from a small variety of options. Language does not work that way. The idea was that based on the list of

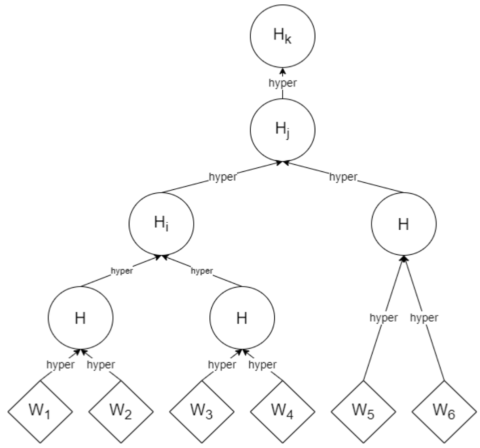

Figure 1: Schematic tree of finding the optimal hypernym as a semantic tag for words W_1-W_6

Subjects the Verb compiled next to itself, I could find their common tag with the use of HuWN.

In HuWN, plenty of information about a word's meanings is coded into synsets, but in the current research we would only need a few of that, namely the lemma (LITERAL) of the given Noun, and the word or expression that is linked to it via the hypernymy relation. Although the transitive nature of hypernymy is not inherently coded into WordNets, and in some cases this can make working with this relation more difficult (Cheng et al., 2023), the way it is represented is enough for us to be able to go from a leaf (in this case: one of the Nouns occurring as Subject) through hypernyms of hypernyms all the way to a root with a recursive algorithm.

It is easy to visualize, based on the previous explanation, that we could find common hypernyms of 2 words with this recursive method by finding the first node in their lists of hypernyms where they overlap with each other. In this case, theoretically, that first overlapping node would be the perfect dynamic tag for the list of the given two words. Expanding on this idea is how we find the possible solution to the current problem. We are given a list of Nouns, we find all their "ancestors", compare these lists, and find the fist common node they share – that is the dynamic tag that can theoretically be the selectional category the Verb places on its Subject. In Figure 1, you can see this concept visualized. The aim is to find H_j, while avoiding the suboptimal H_i and H_k in the process.

In some instances, H_i is the optimal tag we are looking for. When can this be useful? In cases where some words of the wordlist are clear outliers, we can exclude them, and they don't spoil the tag we get by trying to find a tag that would apply to every member of the list. For example, in idioms it is usual that the Subject of a Verb can be a Noun that would not appear in that environment in normal circumstances, like in the expression "jön még kutyára dér" (literally: "the hoarfrost is still to come for the dog", meaning: be sure your sins will find you out), *dér* (hoarfrost) is an unusual Subject for *jön* (come), so this way it could be excluded easily. This might also help to account for different meanings of the same Verb, although the Subjects that belong to other meanings cannot be considered outliers, with the correct scope they could also be filtered out or looked at in another round of tag searching. Staying with the example above, *jön* can mean a physical movement in space or a metaphorical movement of an event in time, and these stand with vastly different Subjects.

Similar methods are widely used in different Natural Language Processing tasks, such as Word Sense Disambiguation (i.e. Resnik, 1995; 1997; Ye, 2004; Dhungana and Shakya, 2015), and improvements on HuWN itself happened with a similar approach (Miháltz et al., 2013), but in the field of the current research it is less utilized. As previously mentioned, this is first and foremost is an attempt for automatization in the pre-processing for the main clustering task, but it would greatly simplify and improve the work process.

The algorithm constructed to solve this task would get a list of Nouns, it would find the leaves each of these nouns are represented on, in that synset it would identify the hypernyms ID under the ILR tag, and using that ID recursively steps up to the root of the hypernymy hierarchy. These lists are then concatenated into a single list, and we could find the optimal tag at the node they all first overlap at. The realization of this code is expanded upon in the next chapter and can be found on GitHub[1].

4 Algorithm

Given that HuWN is accessible in XML file format, a simple code using Python ElementTree (2024) should be the easiest way to find both the full lists of hypernyms to wordlists, and their lowest common hypernym acting as their selectional tag as well. Since each synset contains both a LITERAL and an ID of a meaning, identifying words in HuWN shouldn't be a problem. Moreover, the hypernyms are coded into the synsets using its ID, so finding their representations and their hypernyms can be done using analogous methods.

	id_lit_dict	lit_id_dict	id_hypo_dict
key	ID	LITERAL	ID
value	LITERAL	ID	ID of hypernym

Table 1: Keys and values of the different utilized

There were two attempts of the program that realized the theoretical mechanism outlined in the previous chapter. The first attempt was constructed with the use of ElementTree and the original XML file, but it was a faulty method, because the use of the original file as an outside object made it too slow to help the work, it was rather a hindrance, so it quickly became obsolete.

In the previous chapter I mentioned that only a limited number of the encoded information was needed in this task. This realization made the second attempt possible: I could convert the information into an inside object, and the nature of the information made it possible to use various dictionaries for the task. I used the xmltodict (Blech, 2022) Python library for this. Three dictionaries were constructed and used to substitute the XML file, these were id_lit_dict, lit_id_dict and lit_hypo_dict. Information on these dictionaries can be found in Table 1. The numpy (Harris et al., 2020) library was also used in the code to calculate different variables.

As it can be seen in the GitHub directory, there are four functions in this second attempt, I would like to introduce each of them in short.

4.1 findAllHyperonym

The first function was made to create the full list of hypernyms for a word included in HuWN. It takes a synset ID as its input and gives back a set of IDs as its output – this set being the hypernyms of the input synset. It achieves this by searching through the id_hypo_dict for the input ID and finding every hypernym belonging to that. Then it recursively goes through the same process with the found hypernyms, creating the said set in the

[1] https://github.com/anonymous-subs101/gwc2025

process. The set contains every possible hierarchy of which the original synset could be the lowest member of. This function is embedded into the next one.

4.2 ancestors

With this function we can find all the ancestors a list of words has – some of which are presumably common ancestors. It has one argument, that being a list of words, and its output is a list of the individual words and their lists of hypernyms. With the use of the findAllHyperonym function it can create a concise format of the superordination hierarchy.

4.3 commonN

The commonN function is supposed to finds every common ancestor a list of words has. This third function has two inputs: a list of lists created by the ancestors function and a number between 0 and 1. As an output we get a list of lists, each having a synset ID of a hypernym that is common for the wordlist we had and the list of the words that selection tag applies to.

In Chapter 3, I mentioned that in some cases finding a tag that would apply to every appearing Noun would be suboptimal, and in those cases, it is better to find a tag that applies to only part of the wordlist. With the number in the input, we can specify what percentage of the wordlist should the common ancestor apply to, 1 being 100%.

4.4 lowestCommon

Finally, we have our last function, that is meant to execute the main purpose of the whole project, finding the lowest common ancestor that can be applied to the wordlist as the selectional preference tag. It takes the same input as commonN, a list of words and their hypernyms and a number. This function checks if the possible tag we found is the hypernym of any other candidates for being the tag, and it stops when it finds the word (or phrase in some cases) that is the hypernym of the original wordlist but not any other candidates. Its output is a list, because there could be more than one candidate for which this constrain is true.

5 Results, limitations and future work

From a programming point of view, the algorithm works great. The second version identifies the hypernym that can be the appropriate tag almost instantly, so it would seem to be fit for inserting into the preprocessing chain. However, we should also look at the tag it provides.

In some cases, the tag presented is mostly serviceable, like when we input the Hungarian equivalents of the words *teacher*, *student*, *dog*, *cat*, *horse*, *rabbit*, *mouse*, *human* and *mammoth*, we get "(living or once lived entity)". In other cases, the tag seems to be too specific, and not representing well the intuitive categorization, as in *horse*, *dog* and *rabbit* getting the tag ['placental', 'placental mammal', 'true mammal']. This being the tag makes sense, and it is factually correct, but in the mind of a native speaker, this scientific categorization probably does not take place. This could be rectified by defining a list of categories, that does occur in the speakers' understanding, but that goes beyond the goal of the current project. Lastly, there were cases where the algorithm could not find any suitable tags. This comes up with the implementation of Proper Nouns into the research, for example the pair *teacher* and *Péter* had no common hypernyms, when in a real text it can easily occur, that these words denote the same entity, yet the algorithm finds no point where these two intercept.

All this is to say that in certain limits and cases the algorithm is fine, but without the necessary fine-tuning it is not ready to be inserted into the main task it is supposed to be one of the earliest steps of. Finding any tag for the wordlist can be too vague or too specific to have explanatory value concerning the mental processes involved in semantic selection. While the algorithm can be helpful in finding the fitting tags, it is not ready to be an automated step in the linguistic processing chain that is required in the original clustering task. If we were to stay with the current state, it might be a working semi-automated tool to assist annotation, thus making sure that the human errors are limited, and the tagging is faster.

Accounting for the automation is a different topic. For that, we would need extensive fundamental research in mental semantic category representation while language processing, so that we can have a better understanding of the way native speakers use these semantic selectional rules, and so we could better model that with our algorithm. This, however, would be another, far larger project, than this current one. For the time being, the tool can be applied as assistance – rather than the annotator itself.

References

Martin Blech. 2022. xmltodict.

Hadumod Bußmann and Gregory Trauth. 1996. *Routledge dictionary of language and linguistics*. Routledge, London, [Online-ausg.].

Xuyou Cheng, Michael Schlichtkrull, and Guy Emerson. 2023. Are Embedded Potatoes Still Vegetables? On the Limitations of WordNet Embeddings for Lexical Semantics. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8763–8775, Singapore. Association for Computational Linguistics.

Noam Chomsky. 1965. *Aspects of the theory of syntax*.Massachusetts Institute of Technology. Research Laboratory of Electronics. Special technical report. The MIT Press, Cambridge, Massachusetts, 50th Anniversary Edition.

Udaya Raj Dhungana and Subarna Shakya. 2015. Hypernymy in WordNet, Its Role in WSD, and Its Limitations. In *2015 7th International Conference on Computational Intelligence, Communication Systems and Networks*, pages 15–19, Riga, Latvia. IEEE.

Charles R. Harris, K. Jarrod Millman, Stéfan J. Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. Van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández Del Río, Mark Wiebe, Pearu Peterson, et al. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Márton Miháltz, Csaba Hatvani, Judit Kuti, György Szarvas, János Csirik, Gábor Prószéky, and Tamás Váradi. 2008. Methods and Results of the Hungarian WordNet Project. In Attila Tanács, editor, *GWC 2008: Proceedings*, pages 311–320. University of Szeged, Department of Informatics, Szeged.

Márton Miháltz, Bálint Sass, and Balázs Indig. 2013. What Do We Drink? Automatically Extending Hungarian WordNet With Selectional Preference Relations. In pages 105–109, Trento.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Csaba Oravecz, Tamás Váradi, and Bálint Sass. 2014. The Hungarian Gigaword Corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1719–1723. European Language Resources Association, Reykjavik, Iceland.

Python. 2024. xml.etree.ElementTree — The ElementTree XML API.

Philip Resnik. 1993. Semantic classes and syntactic ambiguity. In *Proceedings of the workshop on Human Language Technology - HLT '93*, page 278, Princeton, New Jersey. Association for Computational Linguistics.

Philip Resnik. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy.

Philip Resnik. 1997. Selectional Preference and Sense Disambiguation. In *Tagging Text with Lexical Semantics: Why, What, and How?*

Veronika Vincze, György Szarvas, and Janos Csirik. 2008. Why are wordnets important? In *European Computing Conference. (ECC 08), WSEAS*, pages 316–322. Malta.

Patrick Ye. 2004. Selectional Preference Based Verb Sense Disambiguation Using WordNet. In *Proceedings of the Australasian Language Technology Workshop 2004*, pages 155–162.