Uneek: a Web Tool for Comparative Analysis of Annotated Texts

Per Malm[1], **Malin Ahlberg**[2], **Dan Rosén**[2]
[1]Department of Scandinavian Languages, Uppsala University, Sweden
[2]Språkbanken, University of Gothenburg, Sweden
per.malm@nordiska.uu.se, malin.ahlberg@gu.se, dan.rosen@svenska.gu.se

Abstract

In this paper, we present *Uneek*, a web based linguistic tool that performs set comparison operations on raw or annotated texts. The tool may be used for automatic distributional analysis, and for disambiguating polysemy with a method that we refer to as *semi-automatic uniqueness differentiation* (SUDi). Uneek outputs the intersection and differences between their listed attributes, e.g. POS, dependencies, word forms, frame elements. This makes it an ideal supplement to methods for lumping or splitting in frame development processes. In order to make some of Uneek's functions more clear, we employ SUDi on a small data set containing the polysemous verb *bake*. As of now, Uneek may only run two files at a time, but there are plans to develop the tool so that it may simultaneously operate on multiple files. Finally, we relate the developmental plans for added functionality, to how such functions may support FrameNet work in the future.

Keywords: frame development, distributional method, automated comparative analysis, polysemy disambiguation

1. Introduction

Uneek is a web based linguistic tool that automatizes complex comparative tasks. It takes two input files (txt or xml) and outputs their intersection, and/or the differences between them. This makes Uneek a suitable aid in frame development processes, e.g. to the splitting or lumping schemes described in Ruppenhofer et al. (2016). The benefits of the program is further illustrated in an example of polysemy disambiguation through a method we refer to as *semi-automatic uniqueness differentiation* (SUDi).

There are other approaches available to polysemy disambiguation. For example, one may choose a more statistical approach such as Drouin (2003) using *TermoStat*, a software designed for term extraction that determines the specificity of words in a domain-specific corpus compared to a larger reference corpus. One may also choose a more qualitative approach, e.g. Ruppenhofer et al. (2016) where the disambiguation of a polysemous form is based on the semantic frames they evoke. In this setting, SUDi may be considered a supplementary method to the same problem.

The paper is organized as follows: in Section 2, we present Uneek. Section 3 holds a presentation on how to use Uneek for polysemy disambiguation in SUDi. The final Section 4 contains some closing remarks and plans for future work.

2. Uneek

Uneek is an open source project, and the code is available under the MIT-license.[1] It is a tool for automatically performing distributional analyses in the sense of Harris (1954), where the "distribution of an element will be understood as the sum of all its environments". There are other programs available today that gives a similar result, e.g. *AntConc* (Anthony, 2016) and *Wordsmith* (Scott, 2017). One downside with the former is that it – to our best knowledge – is not currently designed to handle xml tags.[2] Consequently, it only operates on word level. One downside with the latter is that it is developed for Windows OS, and is not compatible with all operating systems. Uneek handles both txt and xml, and is available for online use by any modern web browser without specific OS requirements.

The chief benefit of Uneek lies in its ability to operate on annotated text. There are a number of freely available tools for automatic annotation, e.g. *Sparv*, an easy to use annotation pipeline for various languages (Borin et al., 2016), and *Stanford CoreNLP* (Manning et al., 2014).[3] Uneek may also be used on annotations from the Berkeley FrameNet.[4] Working on the level of annotations also gives the opportunity to compare texts in different languages, given that they have at least one annotation layer in common. This might be of interest when working with language independent frameworks, such as UD (Nivre et al., 2016).

Uneek has three general settings for (i) set comparison operations, (ii) input format, and (iii) shallow syntactic sequencing. These settings are presented in detail below.

There are two set comparison operations, the intersection ($A \bigcap B$), which we refer to as *intersectional analysis*, and the differences ($A - B$ and $B - A$) which we refer to as *uniqueness differentiation*. Uniqueness differentiation is used for SUDi, or other methods where a full account of the differences between two sets is wanted. For instance, consider the sets A and B in example 1a–b below.

(1) a. $A = \{Aegon, forgave, his, goat\}$
b. $B = \{Aegon, hid, his, goat, yesterday\}$

A uniqueness differentiation of the sets in example 1a–b results in the following two sets:

(2) a. $A - B = \{forgave\}$
b. $B - A = \{hid, yesterday\}$

The intersectional analysis may be used for cases where a full account of what the two sets have in common is wanted. It provides the following set:

[1]The code is found at https://github.com/PerMalm/uneek, and the tool at https://uneek-tools.github.io/.

[2]However, this feature is under development: [last checked 11-01-2018] http://www.laurenceanthony.net/software/antconc/.

[3]There are other tools for FN annotation. See *SEMAFOR* for automatic annotation (Das et al., 2010), and *FrameNet Brasil WebTool* for manual annotation (Torrent et al., 2018).

[4]https://framenet.icsi.berkeley.edu/fndrupal/

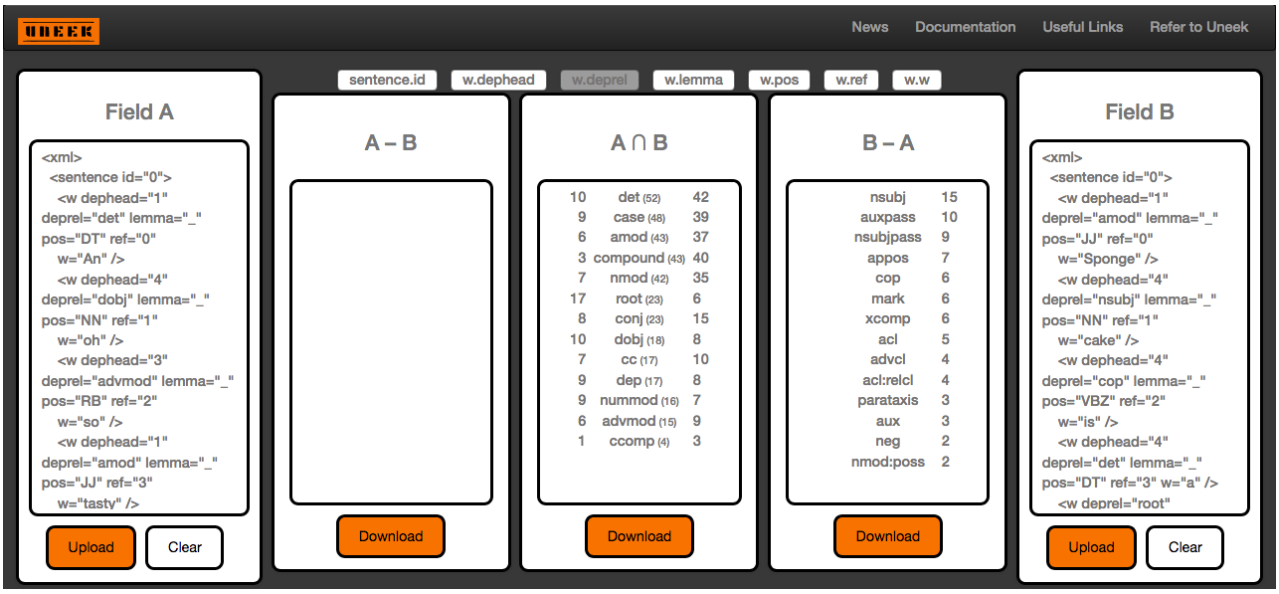

Figure 1: A Uneek analysis of *a recipe for a sponge cake* (Field A) and *a description of sponge cake* (Field B)

(3) $A \bigcap B = \{Aegon, his, goat\}$

Even though these operations are simple, it is helpful to calculate them automatically. To manually perform these tasks on large texts, is both time consuming and error prone. The user chooses between two input formats. If *txt* is chosen, i.e. raw text, a simple whitespace tokenizer is run. The tokenized word forms are then given as input to the set comparisons. If xml format is chosen, all text nodes (typically word forms) as well as all attributes of all tags (typically annotations) are given as input.

A screenshot of Uneek is presented in Figure 1. In the leftmost box labeled Field A, the first input file is uploaded, and the second file is put in the rightmost box (Field B). In this particular case, we have set Uneek to perform an intersectional analysis and a uniqueness differentiation, and uploaded a recipe for sponge cake (File A), and some general description about sponge cake (File B). The input files have been processed using the Stanford Dependency Parser.[5] The result is presented in the three middle boxes. The second rightmost box and the second leftmost box holds the result of the uniqueness differentiation. The middle box shows the result of the intersectional analysis.

The attributes of the xml – corresponding to annotation layers – are visualized as radio buttons above the result boxes. These buttons control which layer is shown in the result box. In Figure 1 we have chosen to look at dependencies, which among other things tells us that the recipe for sponge cake lacks nominal subjects (nsubj). This is to be expected due to the imperative mood of recipes.

The third general setting allows for set comparisons on shallow syntactic sequences. A shallow syntactic sequence is here understood as a left to right organization of linguistic units specified in the xml. The only assumption made for the shallow syntax function is that the xml tag `sentence` sets the span in which the syntax chains are constructed. Below, we show some syntactic sequences in various annotation layers for example 1a.

(4) Text $=\{Aegon, forgave, his, goat\}$
Dep $=\{$nsubj, root, nmod:poss, dobj$\}$
Frame $=\{$Forgiveness$\}$
Frame Element $=\{$Judge, Evaluee, [INI]$\}$

The application of set operations on the syntactic sequences returns all the unique syntactic configurations for File A and File B, and all of their shared configurations. This function is especially useful for lexico-grammatical purposes. One may quite easily get a complete account of all the combinatorial possibilities of surface forms for complex constructions and frames. For instance, this function would probably ease the lexicographic frame annotation mode mentioned in Ruppenhofer et al. (2016, 19) by automatically listing all combinations of frame elements (FEs). To minimize visual clutter, the GUI only provides descriptive statistics (raw numbers). The output data may be downloaded in a human and machine readable format (csv) to ease export to statistical programs.

In sum: Uneek operates on user defined data, either raw or annotated text, and provides formal support for intuitions on lumping or splitting linguistic units. It may be used for automatic distributional analysis or for the disambiguation of polysemy presented next.

3. Semi-automatic Uniqueness Differentiation

The rationale for SUDi and Uneek rest on the distributional hypothesis (Harris, 1954) and set theory; see *locus classicus* Cantor (1915). Regarding the former, Firth (1962) wrote, ”You shall know a word by the company it keeps”. However, for the treatment of polysemy we assume: you shall know the difference between two polysemous words by the company one of them constantly rejects. In this section, we present the details of our proposed method for pol-

[5] StanfordDependencyParser from nltk.parse.stanford, v. 3.2.4.

ysemy disambiguation, called semi-automatic uniqueness differentiation (SUDi).

As we indicated in the previous section, there are two reasons for developing Uneek and SUDi. The first is to facilitate finding formal support for linguistic intuitions in complex material. The second is to improve the reliability of the distributional analysis. SUDi is a formalized and semi-automized methodology of some of the work that linguists often do: collect data, sort it and look for differences.

Roughly, SUDi involves five steps: (i) collect cases containing the presumed polysemous form from a corpus, (ii) sort these intuitively into two text-files (iii) process the files using an annotation device that produces xml which is needed in the next step, (iv) run the xml-files through Uneek, and (v) interpret the result.

Using Uneek in step (iv) not only speeds things up, but also simplifies reproducibility. However, to ensure validity, the user should (among other things) delimit the specified environment for the polysemous form in the input data. As for any tool, garbage in results in garbage out.

If Uneek does not find unique forms for one of the files, there is no formal support for polysemy. But, if it does, a linguist needs to interpret the result.

Step (v) in SUDi is based on proof by contradiction using human grammaticality judgements. First, take the linguistic unit that is unique in one of the files, and place it in the context of the polysemous item in the other file. If this switch leads to a semantic change that is deemed unfit in the tested domain (here marked with #), then there is *positive* formal support to the intuition that the polysemous form may be split into different frames, constructions, and so on. Though, if the linguistic unit works fine in the other context, then there is *negative* formal support for polysemy. Being unique in one domain does not lead to its infelicity in the other; uniqueness must be validated. Let us illustrate this step with an example case for which we strongly expect positive support for polysemy, namely for the verb *bake*.

First, we collect example sentences for *bake* from the Berkeley FrameNet COOKING CREATION frame and for *bake* from the APPLY HEAT frame.[6] Second, we sort these in two files. Third, we automatically process them (again using the Stanford Dependency Parser). Fourth, we run the files through Uneek, and interpret the unique differences using the method in step five. The result of step four is presented in Table 1 where only some of the unique values for the COOKING CREATION *bake.v* are given.

Table 1: Unique values for *bake* (COOKING CREATION)

ATTRIBUTES	VALUES (in absolute numbers)
DEP-HEADS:	auxiliary (10), predeterminer (2)
POS:	modal verb (5), poss. pronoun (4)
WORDS:	*Sunday* (2), *cakes* (1), *Saturday* (1)

Recall the uniqueness differentiation between the description and the recipe of sponge cake. Here we expect similar results to support that the difference between the COOKING CREATION and the APPLY HEAT frames, lies in the latter being more recipe-like. For instance, the unique distribution with auxiliaries and modals in Table 1 is explained by the fact that recipes are written in the imperative mood.

Next, we test some of the unique values in Table 1 against a Berkeley FrameNet example from the APPLY HEAT frame, i.e. *Bake the soufflés for 12 minutes*. These tests are presented in example 5a–d below.

(5) a. # Bake *all* the soufflés for 12 minutes.
b. # Bake *your* soufflés for 12 minutes.
c. Bake the soufflés $\left\{\begin{matrix}\text{\# on } \textit{Saturday} \\ \text{for 12 minutes}\end{matrix}\right\}$.
d. Bake the *cakes* for 12 minutes.

From example 5a, we notice that *all* does not fit very well; it may be hard to find recipes for multi-soufflé cooking. Another rare bird in the recipe genre is to state the owner of the soufflé, as in example 5b, so is the instruction of cooking on specific weekdays (example 5c). On the contrary, these words work well in the COOKING CREATION frame, e.g. *Don't worry darling! I'll bake all your soufflés tomorrow.* However, observe that the unique form *cakes* (example 5d) can occur in the APPLY HEAT frame. Hence, it is important to manually interpret the unique units, especially with the open word classes being what they are, i.e. open.

As a corroborative digression, we apply SUDi on the FEs in the annotated sentences for the COOKING CREATION and the APPLY HEAT frame. The result is shown in Table 2.

Table 2: Unique and shared frame elements for *bake* in the COOKING CREATION (A) and the APPLY HEAT frame (B)

$A - B$	$A \bigcap B$	$B - A$
Ingredients, Place, Recipient, Time, Produced food, Purpose	Target Cook Food	Temperature setting, Heating instrument, Duration, Manner, Container

Among other things, we note that the unique FEs Recipient and Time support the observations that were based on example 5b–c above. In conclusion: auxiliaries, possessives, and predeterminers indicate positive formal support for polysemy. But keep in mind the scarce input (36 sentences).

Speed and reliability of automized linguistic labour must not come at any cost, especially not at the price of validity. One should think twice before taking the human element out of the equation. A similar point is made in Fillmore (1992) about the pitfall of exclusively relying on intuitive data or empirical data, a point he makes clear by the following interaction between two radicalized linguists:

> [. . .] the corpus linguist says to the armchair linguist, "Why should I think that what you tell me is true?", and the armchair linguist says to the corpus linguist, "Why should I think that what you tell me is interesting?" Fillmore (1992)

[6] https://framenet.icsi.berkeley.edu/fndrupal/

Fillmore argues for the need of both these radicals, i.e. a computer aided armchair linguist, who checks his/her grammaticality judgements against a corpus in some organized fashion. Still, even behind results that are ever so true and interesting, methodological problems may sometimes lurk about unnoticed, especially in manually performed distributional analyses. When faced with such cases, it is sensible to ask: why should I think that what you tell me is based on a reliable method? A true and interesting result does not necessarily paint the whole distributional picture. We want to be able to say that given a specific corpus and a specific method, we will always get the complete distribution of a particular linguistic unit. We believe that Uneek and SUDi allows linguists to make such statements.

4. Closing Remarks and Future Work

We have presented Uneek, some of its functions, and its potential to mitigate some of the methodological sufferings of linguistic labor. However, we see plenty of room for improvement. Here, we briefly mention two upcoming practical additions to Uneek: (i) *syntactic scope*, and (ii) a *multiple set analysis*.

(i) Sometimes, while faced with complex material, one would like to single out specific constituents of the parse tree for analysis, e.g. the subject. We plan to add functionality to Uneek to automatically extract these constituents. The user should be able to choose a constituent and get the annotation layers for its daughter nodes. This would considerably lessen the preprocessing of the input data.

(ii) We also plan to add a multiple set analys, enabling the user to get the intersection and difference between two or more sets. This would enable researchers and students to get results for complex comparative linguistic studies. Such an addition could come in handy soon, with the Multilingual FrameNet (MLFN) project underway. At the end of this project, Uneek could be used to answer some of the general MLFN questions below.[7]

1. "Are some frames universal?"
2. "Are there regular patterns of differences based on language families, regional groupings, etc.?"

The first question could then be answered by a multiple set intersectional analys of the annotation layers of language specific FrameNets. Uneek would automatically return their shared elements (frames, FEs, phrases, and so on). The second question may be answered by a multiple set uniqueness differentiation on sets of the FNs. Again, it would automatically return their unique elements.

Uneek is a simple tool, but sometimes there is strength in simplicity. Hopefully it will make the processing of complex data less tedious, enabling linguists to focus on the more interesting part of the field, i.e. coming up with explanations for linguistic phenomena.

5. Acknowledgements

The work presented here has been financially supported by the Swedish Research Council through its funding of the projects *South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics* (2015–2019; contract 421-2014-969) and *Swe-Clarin* (2014–2018; contract 821-2013-2003), the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) through its Swedish-Brazilian research collaboration program (2014–2019; contract BR2014-5860), and the University of Gothenburg, its Faculty of Arts and its Department of Swedish.

6. Bibliographical References

Anthony, L. (2016). AntConc (version 3.4.4)[computer software]. Available at http://www. laurenceanthony.net Tokyo, Japan: Waseda University.

Borin, L., Forsberg, M., Hammarstedt, M., Rosén, D., Schäfer, R., and Schumacher, A. (2016). Sparv: Språkbanken's corpus annotation pipeline infrastructure. In *SLTC 2016. The Sixth Swedish Language Technology Conference, Umeå University, 17–18 November, 2016*.

Cantor, G. (1915). *Contributions to the Founding of the Theory of Transfinite Numbers*. Number 1. Open Court Publishing Company.

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010). SEMAFOR 1.0: A Probabilistic Frame-Semantic Parser. *Language Technologies Institute, School of Computer Science, Carnegie Mellon University*.

Drouin, P. (2003). Term extraction using non-technical corpora as a point of leverage. *Terminology*, 9(1):99–115.

Fillmore, C. J. (1992). "Corpus linguistics" vs. "computer-aided armchair linguistics". In *Directions in Corpus Linguistics: Proceedings from a 1991 Nobel Symposium on Corpus Linguistics*, Stockholm. Mouton de Gruyter.

Firth, J. R. (1962). *A Synopsis of Linguistic Theory, 1930-1955*. Basil Blackwell, Oxford, [1957] edition.

Harris, Z. S. (1954). Distributional Structure. *Word*, 10(2-3):146–162.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R. T., Petrov, S., Pyysalo, S., Silveira, N., et al. (2016). Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Ruppenhofer, J., Ellsworth, M., Petruck, M. R., Johnson, C. R., Baker, C. F., and Scheffczyk, J. (2016). FrameNet II: Extended Theory and Practice.

Scott, M. (2017). WordSmith Tools Help. http://www.lexically.net/downloads/version7/HTML/overview.html.

Torrent, T. T., da Silva Matos, E. E., Sigiliano, N. S., da Costa, A. D., and de Almeida, V. G. (2018). A flexible tool for an enriched FrameNet: the FrameNet Brasil Webtool. submitted for publication.

[7] https://framenet.icsi.berkeley.edu/fndrupal/node/5549