Simplified Dependency Annotations with GFL-Web

Michael T. Mordowanec Nathan Schneider Chris Dyer Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
michael.mordowanec@gmail.com, {nschneid,cdyer,nasmith}@cs.cmu.edu

Abstract

We present GFL-Web, a web-based interface for syntactic dependency annotation with the lightweight FUDG/GFL formalism. Syntactic attachments are specified in GFL notation and visualized as a graph. A one-day pilot of this workflow with 26 annotators established that even novices were, with a bit of training, able to rapidly annotate the syntax of English Twitter messages. The open-source tool is easily installed and configured; it is available at: https://github.com/Mordeaux/gfl_web

1 Introduction

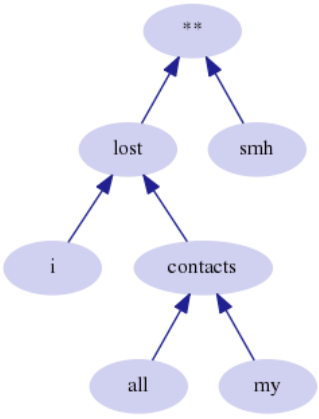
High-quality syntactic annotation of natural language is expensive to produce. Well-known large-scale syntactic annotation projects, such as the Penn Treebank (Marcus et al., 1993), the English Web Treebank (Bies et al., 2012), the Penn Arabic Treebank (Maamouri et al., 2004), and the Prague dependency treebanks (Hajič, 1998; Čmejrek et al., 2005), have relied on expert linguists to produce carefully-controlled annotated data. Because this process is costly, such annotation projects have been undertaken for only a handful of important languages. Therefore, developing syntactic resources for less-studied, lower-resource, or politically less important languages and genres will require alternative methods. To address this, simplified annotation schemes that trade cost for detail have been proposed (Habash and Roth, 2009).[1]

[1] These can be especially effective when some details of the syntax can be predicted automatically with high accuracy (Alkuhlani et al., 2013).

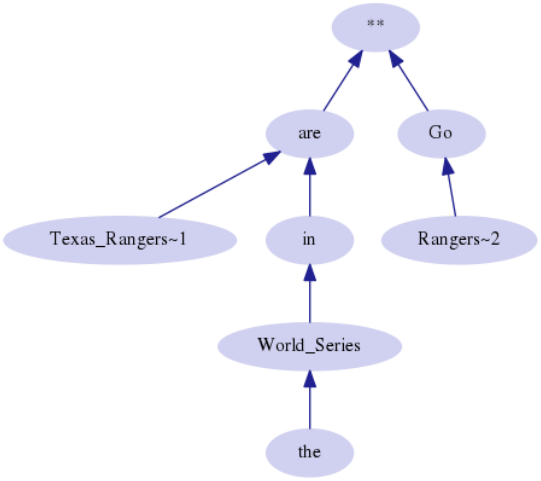
The Fragmentary Unlabeled Dependency Grammar (FUDG) formalism (Schneider et al., 2013) was proposed as a simplified framework for annotating dependency syntax. Annotation effort is reduced by relaxing a number of constraints placed on traditional annotators: partial fragments can be specified where the annotator is uncertain of part of the structure or wishes to focus only on certain phenomena (such as verbal argument structure). FUDG also offers mechanisms for excluding extraneous tokens from the annotation, for marking multiword units, and for describing coordinate structures. FUDG is written in an ASCII-based notation for annotations called Graph Fragment Language (GFL), and text-based tools for verifying, converting, and rendering GFL annotations are provided.

Although GFL offers a number of conveniences to annotators, the text-based UI is limiting: the existing tools require constant switching between a text editor and executing commands, and there are no tools for managing a large-scale annotation effort. Additionally, user interface research has found marked preferences for and better performance with graphical tools relative to text-based interfaces—particularly for less computer-savvy users (Staggers and Kobus, 2000). In this paper, we present the **GFL-Web** tool, a web-based interface for FUDG/GFL annotation. The simple interface provides instantaneous feedback on the well-formedness of a GFL annotation, and by wrapping Schneider et al.'s notation parsing and rendering software, gives a user-friendly visualization of the annotated sentence. The tool itself is lightweight, multi-user, and easily deployed with few software dependencies. Sentences are assigned to annotators via an administrative interface, which also records progress and provides for a text dump of

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 121–126,
Baltimore, Maryland USA, June 23-24, 2014.



(a) @Bryan_wright11 i lost all my contacts , smh .



(b) Texas Rangers are in the World Series ! Go Rangers !!!!!!!!! http://fb.me/D2LsXBJx

Figure 1: FUDG annotation graphs for two tweets.

all annotations. The interface for annotators is designed to be as simple as possible.

We provide an overview of the FUDG/GFL framework (§2), detail how the tool is set up and utilized (§3), and discuss a pilot exercise in which 26 users provided nearly 1,000 annotations of English Twitter messages (§4). Finally, we note some of the technical aspects of the tool (§5) and related syntactic annotation software (§6).

2 Background

GFL-Web is designed to simplify the creation of dependency treebanks from noisy or under-resourced data; to that end, it exploits the lightweight FUDG/GFL framework of Schneider et al. (2013). Here we outline how FUDG differs from traditional Dependency Grammar (§2.1) and detail major aspects of GFL (§2.2).

2.1 FUDG

Figure 1 displays two FUDG graphs of annotations of Twitter messages ("tweets", shown below in tokenized form). Arrows point upwards from dependents to their heads. These tweets illustrate several characteristics of the formalism, including:

- The input may contain multiple independent syntactic units, or "utterances"; the annotation indicates these by attaching their heads to a special root node called `**`.
- Some input tokens are omitted if deemed extrinsic to the syntax; by convention, these include most punctuation, hashtags, usernames, and URLs.
- Multiword units may be joined to form composite lexical nodes (e.g., `World_Series` in figure 1b). These nodes are not annotated with any internal syntactic structure.
- Tokens that are used in the FUDG parse must be unambiguous. If a word appears multiple times in the input, it is disambiguated with `~` and an index (e.g., `Rangers~2` in figure 1b).

(Some of the other mechanisms in FUDG, such as coordinate structures and underspecification, are not shown here; they are not important for purposes of this paper.)

2.2 GFL

The Graph Fragment Language is a simple ASCII-based language for FUDG annotations. Its norms are designed to be familiar to users with basic programming language proficiency, and they are intuitive and easy to learn even for those without. The annotation in figure 1a may be expressed in GFL as:[2]

```
i > lost** < ({all my} > contacts)
smh**
```

In GFL, angle brackets point from a dependent (child) to its head (parent). Parentheses group nodes together; the head of this group is then attached to another node. The double asterisk (`**`) marks a root node in an annotations containing multiple utterances. Curly braces group nodes that modify the same head.

GFL corresponding to Figure 1b is:

[2]The abbreviation smh stands for *shaking my head*.

Sentence: Texas Rangers are in the World Series ! Go Rangers !!!!!!!!! http://fb.me/D2LsXBJx

Input format:

```
---
% ID data_set_name:417
% TEXT
Texas Rangers~1 are in the World Series ! Go Rangers~2 !!!!!!!!!
http://fb.me/D2LsXBJx
% ANNO
Texas Rangers~1 are in the World Series  Go Rangers~2
http://fb.me/D2LsXBJx
```

Figure 2: Illustration of the GFL-Web input format for a tweet. The ANNO section will be shown to the user as the default annotation; punctuation has been stripped out automatically to save time.

Figure 3: User home screen showing assigned batches for annotation, with links to the training set and blank annotation form.

```
[Texas Rangers~1] > are** < in
in < (the > [World Series])
Go** < Rangers~2
```

This uses square brackets for multiword expressions. Similar to a programming language, there are often many equivalent GFL annotation options for a given sentence. The annotation can be split across multiple lines so that annotators can approach smaller units first and then link them together.

3 Using GFL-Web

The GFL-Web tool uses the Python programming language's Flask microframework for server-side scripting. This allows it to be deployed on a web server, locally or via the Internet. This also enables the interface to rely on scripts previously created for analyzing GFL. Once installed, the researcher need only configure a few settings and begin entering data to be annotated.

3.1 Setup

There are a few simple configuration options. The most useful of these options specify how many sentences should be in each batch that is assigned to an annotator, and how many sentences in each batch should be doubly annotated, for the purpose of assessing inter-annotator agreement. By default, the batch size is 10, and the first 2 sentences of each batch overlap with the previous batch, so 4/10 of the sentences in the batch will be annotated by someone else (assuming no two consecutive batches are assigned to the same user). The program requires tokenized input, with indices added to distinguish between words that appear twice (easily automated). The input format, figure 2, allows for easy editing with a text editor if so desired.

Once the input files have been placed in a designated directory, an **admin interface** can be used to assign batches of data to specific users (annotators).

3.2 Annotation

Annotators log in with their username and see a **home screen**, figure 3. The home screen always offers links to a **training set** to get them up to speed, as well as a **blank** annotation form into which they can enter and annotate any sentence. Beneath these is a table of batches of sentences which have been assigned to the user. Clicking

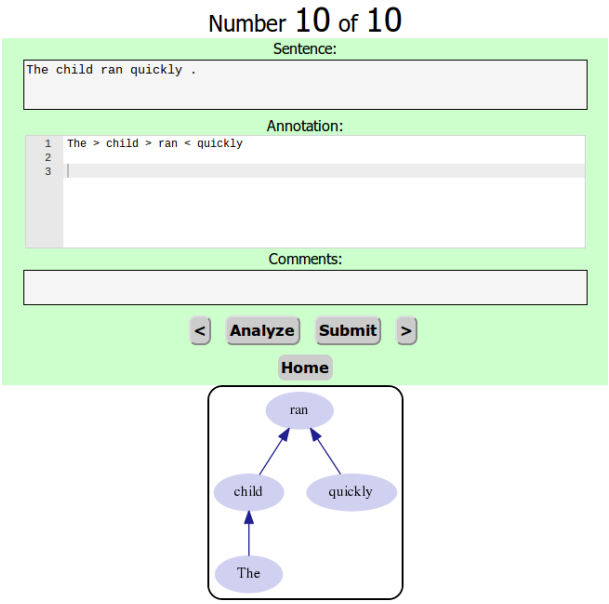


Figure 4: A well-formed GFL annotation is indicated by a green background and visualization of the analysis graph.

any of these will take the annotator to an annotation page, which displays the text to be annotated, an input field, and a comments box.

The annotation interface is simple and intuitive and provides instant feedback, preventing the annotator from submitting ill-formed annotations. Annotators press the Analyze button and receive feedback before submitting annotations (figure 4). Common GFL errors such as unbalanced parentheses are caught by the program and brought to the attention of the annotator with an informative error message (figure 5). The annotator can then fix the error, and will be able to submit once all errors are resolved.

The training set consists of 15 sentences selected from Rossman and Mills (1922), shown in the same annotation interface. Examples become increasingly more complicated in order to familiarize the user with different syntactic phenomena and the entry-analyze-review workflow. A button displays the FUDG graph from an expert annotation so the novice can compare it to her own and consult the guidelines (or ask for help) where the two graphs deviate.

4 Pilot User Study

We conducted a pilot annotation project in which 26 annotators were trained on GFL-Web and asked to annotate English Twitter messages from the *daily547* and *oct27* Twitter datasets of Gimpel et al. (2011). The overwhelming majority were all trained on the same day, having no prior knowledge of GFL. Most, but not all, were native speakers of English. Those who had no prior knowledge of dependency grammar in general received a short tutorial on the fundamentals before being introduced to the annotation workflow. All participants who were new to FUDG/GFL worked through the training set before moving on to the Twitter data. Annotators were furnished with the English annotation guidelines of Schneider et al. (2013).[3]

4.1 Results

In the one-day event, 906 annotations were generated. Inter-annotator agreement was high—.9 according to the *softComPrec* measure (Schneider et al., 2013)—and an expert's examination of a sample of the annotations found that 75% of contained no major error.

Annotators used the analysis feature of the interface—which displays either a visual representation of the tree or an error message—an average of 3.06 times per annotation. The interface requires they analyze each annotation at least once. Annotators have the ability to resubmit annotations if they later realize they made an error, and each annotation was submitted an average of 1.16 times. Disregarding instances that took over 1,000 seconds (under the assumption that these represent annotators taking breaks), the median time between the first analysis and the first submission of an annotation was 30 seconds. We take this as evidence that annotators found the instant feedback features useful in refining their annotations.

4.2 Post-Pilot Improvements

Annotator feedback prompted some changes to the interface. The annotation input box was changed to incorporate bracket-matching. The graph visualization for a correct annotation was added for each example in the training set so new annotators could compare their tree to an example. Presumably these changes would further reduce annotators' training time and improve their efficiency. Progress bars were added to the user home screen to show per-batch completion information.

4.3 Other Languages

In addition to English, guidelines for Swahili, Zulu, and Mandarin are currently in development.

[3] https://github.com/brendano/gfl_syntax/blob/master/guidelines/guidelines.md

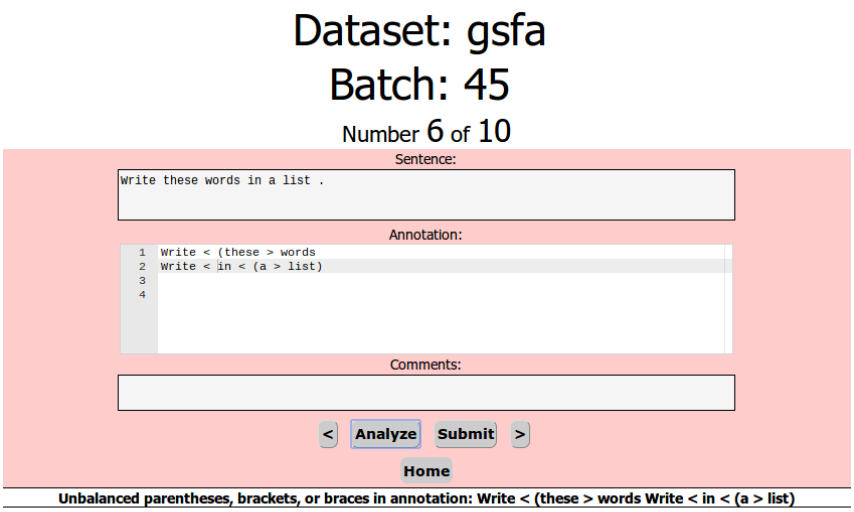


Figure 5: An example error notification. The red background indicates an error, and the cause of the error is displayed at the bottom of the screen.

5 Technical Architecture

GFL-Web and its software dependencies for analyzing and visualizing GFL are largely written in Python. The tool is built with Flask, a Python framework for web applications. Data is stored and transmitted to and from the browser in the Javascript Object Notation (JSON) format, which is supported by libraries in most programming languages. The browser interface uses AJAX techniques to interact dynamically with the server.

GFL-Web is written for Python version 2.7. It wraps scripts previously written for the analysis and visualization of GFL (Schneider et al., 2013). These in turn require Graphviz (Ellson et al., 2002), which is freely available.

Flask provides a built-in server, but can also be deployed in Apache, via WSGI or CGI, etc.

6 Other Tools

In treebanking, a good user interface is essential for annotator productivity and accuracy. Several existing tools support dependency annotation; GFL-Web is the first designed specifically for the FUDG/GFL framework. Some, including WebAnno (Yimam et al., 2013) and brat (Stenetorp et al., 2012), are browser-based, while WordFreak (Morton and LaCivita, 2003), Abar-Hitz (Ilarraza et al., 2004), and TrEd (Pajas and Fabian, 2000–2011) are client-side applications. All offer tree visualizations; to the best of our knowledge, ours is the only dependency annotation interface that has text as the exclusive mode of entry. Some, such as WebAnno and brat, aim to be fairly general-purpose, supporting a wide range of annotation schemes; by contrast, GFL-Web supports a single annotation scheme, which keeps the configuration (and codebase) simple. In the future, GFL-Web might incorporate elements of monitoring progress, such as display of evaluation measures computed with existing FUDG/GFL scripts.

Certain elements of the FUDG/GFL framework can be found in other annotation systems, such as the PASSAGE syntactic representation (Vilnat et al., 2010), which allows for grouping of words into units, but still requires dependency relations to be labeled.

Finally, we note that new approaches to corpus annotation of *semantic* dependencies also come with rich browser-based annotation interfaces (Banarescu et al., 2013; Abend and Rappoport, 2013).

7 Conclusion

While the creation of high-quality, highly specified, syntactically annotated corpora is a goal that is out of reach for most languages and genres, GFL-Web facilitates a rapid annotation workflow within a simple framework for dependency syntax. More information on FUDG/GFL is available at `http://www.ark.cs.cmu.edu/FUDG/`, and the source code for GFL-Web is available at `https://github.com/Mordeaux/gfl_web`.

Acknowledgments

The authors thank Archna Bhatia, Lori Levin, Jason Baldridge, Dan Garrette, Jason Mielens, Liang Sun, Shay Cohen, Spencer Onuffer, Nora Kazour, Manaal Faruqui, Wang Ling, Waleed Ammar, David Bamman, Dallas Card, Jeff Flanigan, Lingpeng Kong, Bill McDowell, Brendan O'Connor, Tobi Owoputi, Yanchuan Sim, Swabha Swayamdipta, and Dani Yogatama for annotating data, and anonymous reviewers for helpful feedback. This research was supported by NSF grant IIS-1352440.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238. Sofia, Bulgaria.

Sarah Alkuhlani, Nizar Habash, and Ryan Roth. 2013. Automatic morphological enrichment of a morphologically underspecified treebank. In *Proc. of NAACL-HLT*, pages 460–470. Atlanta, Georgia.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186. Sofia, Bulgaria.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, PA.

Martin Čmejrek, Jan Cuřín, Jan Hajič, and Jiří Havelka. 2005. Prague Czech-English Dependency Treebank: resource for structure-based MT. In *Proc. of EAMT*, pages 73–78. Budapest, Hungary.

John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. 2002. Graphviz—open source graph drawing tools. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing*, pages 483–484. Springer, Berlin.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proc. of ACL-HLT*, pages 42–47. Portland, Oregon.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proc. of ACL-IJCNLP*, pages 221–224. Suntec, Singapore.

Jan Hajič. 1998. Building a syntactically annotated corpus: the Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press, Prague.

Arantza Díaz De Ilarraza, Aitzpea Garmendia, and Maite Oronoz. 2004. Abar-Hitz: An annotation tool for the Basque dependency treebank. In *Proc. of LREC*, pages 251–254. Lisbon, Portugal.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109. Cairo.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Thomas Morton and Jeremy LaCivita. 2003. WordFreak: An open tool for linguistic annotation. In *Proc. of HLT-NAACL: Demonstrations*, pages 17–18. Edmonton, Canada.

Petr Pajas and Peter Fabian. 2000–2011. Tree Editor TrEd 2.0. `http://ufal.mff.cuni.cz/tred/`.

Mary Blanche Rossman and Mary Wilda Mills. 1922. *Graded Sentences for Analysis, Selected from the Best Literature and Systematically Graded for Class Use*. L. A. Noble.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 51–60. Sofia, Bulgaria.

Nancy Staggers and David Kobus. 2000. Comparing response time, errors, and satisfaction between text-based and graphical user interfaces during nursing order tasks. *Journal of the American Medical Informatics Association*, 7(2):164–176.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proc. of EACL: Demonstrations*, pages 102–107. Avignon, France.

Anne Vilnat, Patrick Paroubek, Eric Villemonte de la Clergerie, Gil Francopoulo, and Marie-Laure Guénot. 2010. PASSAGE syntactic representation: a minimal common ground for evaluation. In *Proc. of LREC*, pages 2478–2485. Valletta, Malta.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proc. of ACL: Demonstrations*, pages 1–6. Sofia, Bulgaria.