Honing the Sketch Engine Prepositions

Ken Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872 USA
ken@clres.com

Abstract

The initial construction of the Sketch Engine corpus for the Pattern Dictionary of English Prepositions (PDEP) contained 99 percent of the instances. We refined the original Python script used to create the SE files to understand the difficulties with the missing instances and make further examination of problematic identification of the complements and governors in other instances. After making these refinements, we have updated the SE corpus. In addition, we have incorporated these refinements in the PDEP improvements in the files that can be downloaded to interested parties. The original script focuses on the readily processing of data in PDEP parses and feature files associated to the preposition instances. Here, we modified the script to output messages and problems describing the processing for each instance. (Here.)

1. Introduction

Litkowski (2017) describes the Pattern Dictionary of English Prepositions (PDEP, Litkowski (2014)) and the procedures used to create the files to be installed into Sketch Engine[1] (SE). Litkowski (2017) also describes how to use SE for the preposition corpus. That paper also describes, in its appendix, the Python script and all its subroutines. In section 2, we describe the changes in the Python script, particularly how the changes report on each instance and provide the basis for examining the results. Section 3 summarizes the effects on each corpus. Section 4 describes how this script affects on the PDEP data available to users.

2. Changes in the SE Script to Create Vertical Files

In the initial implementation[2] of the **download_parses.py**, processing wrote some information about the instances to the standard error (**stderr**) or standard output (**stdout**). This information was not saved, but this output provided an opportunity for a more detailed examination of the processes.

The main function for the script was to create vertical files for the parses for each instance available at PDEP. These parses, using the dependency CoNLL-X format, did not include certain basic data about each instance, specifically, the preposition name, the corpus, the preposition sense, and the corpus instance number. We also add information about the class and the subclass for the preposition sense, the lemma of the complement and the governor, and a supersense tag for the complement and the governor.

The **main** function gets the complement and the governor data and the parses for each preposition. There is no substantive difference in this function, except printing to a message file identification for the

[1] https://www.sketchengine.eu/
[2] The herculean and innovative efforts for this script were provided by Marek Medved at Lexical Computing, Ltd.

preposition, using the same initial line for the preposition, i.e., the XML **doc** element with the corpus name and the preposition file name. This function contains the primary function, **get_content**, which processes the parse file for each preposition.

The essence of **get_content** is chiefly the same, first getting the JSON complement and governor data for each instance (printing this data in a message file), then getting each sentence (involving linking it to its parse), and finally processing the file for the tokenized sentence into the vertical file (with the function **create_new_vert**). This function has three minor differences. (1) Some instances are unable to find the complement or the governor. In the original, if the governor could not be found, no attempt was made to search for the complement; the modification tried for each, rather than in one simultaneous attempt. This modification obtained some instances for which a complement could be found. (2) Part of the process for linking the sentence instance with the complement-governor instance made a call of **prepsents.php** for each sense. This script could call for "all" senses than each one (e.g., one call rather than 250 calls). This modification was simply a little bit more efficient. (3) In some cases, it was not possible to find a matching sentence for the specified sense. This fact was part of **stderr**, but otherwise not recorded. This fact was also printed out for the problem file, for use for further examination of the reason.

The function **create_new_vert** continued the procedure for adding each token using the parse, i.e., the token identification, the token word, the lemma (with a single letter part of speech), the part of speech, the dependency identification, and the syntactic type. Before the tokens were entered into the vertical file, the function **include_component** was the first step in this function to identify the starting and the ending location of the preposition, the complement, and the governor. The function was modified in two ways: (1) adding XML elements and their attributes to the vertical file and (2) recording problems in not being able to close elements for an instance. A **<s>** element was entered before any tokens were added to the vertical file for a sentence. Attributes were added to the **<s>** element for the preposition sense (**sense_label**), the class of the sense (**class**), the subclass (**subc**) if available, the instance number (**inst**), and a link to allow a user to examine the PDEP pattern fields.[3] The function continued putting opening and ending element names before and after the token in the vertical file (**<prep>**, **<compl>**, and **<gov>**). In the modification, a supersense tag attribute (**sst**) was added to the complement and the governor tokens when one could be identified. The method of assigning a supersense used the WordNet lexicographer file name, as in McCarthy et al., (2015). The second change of this function recorded a problem in finding the ending element of the preposition (**</prep>**), the complement (**</compl>**), or the governor (**</gov>**).

The function **include_component** tries to find the starting and the ending token for the preposition, the complement, and the governor. To accomplish this, the important argument is the "plain sentence" containing the objective structure. In some cases, an empty "plain sentence" was the value of the argument. In other cases, it was not possible to begin and/or end the desired structure (i.e., the preposition, the complement, or the governor). In these cases, the function returned the value "-1" for the start and the end as the result of the function. There was no modification of this functionality, but rather we wrote a line to the problem file, indicating the corpus, the preposition, the sense, the instance, and the structure that couldn't be found.

No other functions were modified. In summary, only a few minor changes were made to the script content. The primary changes were printing out to identify the instances that could not be fully

[3] E.g., http://clres.com/db/TPPpattern.php?prep=abaft&sense=1(1)

characterized. By doing this, we were able to allow us to summarize statistics about the effectiveness of the script and to identify the problematic instances.

3. Characterizing Instances in the PDEP Vertical Files

The script to create the PDEP vertical file for SE involved three runs, once for each subcorpus: the Corpus Pattern Analysis (CPA/TPP), the Oxford English Corpus (OEC), and the FrameNet Corpus (FN). Details for the script and each corpus are shown in the web pages for "Preposition Corpora in Sketch Engine."[4] The page "Corpora Sources" provides the overall number of the initial instances and the number of instances that were actual included in PDEP. The links in this page provide the details for each of the three corpora. The first web page under each corpus identifies the total number of instances in the corpus and identifies the number of instances that were not included in PDEP. The omitted instances and the reasons for the omissions are described in the other web pages. The changes in the script made it possible to understand the omissions and the improvements in the vertical files.

Each corpus has a web page that describes statistics for the vertical file created by the script. The current set of the runs is now the fourth version and is compared with the third version. The datum is the length of running the **time** for each corpus; the total for all the corpora is about 50 minutes. We list the number of **lines** for each corpus; increases in the number of lines generally corresponds to the ability to identify more elements for complements or governors. Generally, the number of **tokens** was about the same between versions; we improve the tokenization for the OEC and the FN corpora, accounting for the larger number of tokens in these two corpora. The numbers of **sentences** and **prepositions** were generally the same within each version. The number of sentences and prepositions are the same between versions for the CPA/TPP and the OEC corpora, but there are 300 fewer in the earlier version of the FN corpus.

The **complements** were found for approximately 97 percent (78114/80695) of the instances for the three corpora. **Complements with supersenses** were tagged for 85 percent (66145/78114) for the three corpora. The main reasons for the fewer instances with supersenses were the occurrence of pronouns and proper nouns. **Governors** were found for approximately the same as the percentages of the complements, about 96 percent (77792/80695). The percentages of **governors with supersenses** were many fewer instances, again 84 percent (64977/77792), because of pronouns and copular verbs.

As mentioned above in discussing the function **get_content**, in some cases, it was not possible to find a matching sentence for the specified sense, and printed a problem beginning with "Not found for record". We indicated that this would be examined further. These instances are different for the three corpora. For CPA, the full set of the 56 instances for the preposition *vis-à-vis*, for which the features were not generated, possibly because of the hyphen in the name of the preposition. For OEC, there are 27 instances not included in the vertical file, 11 because of a double quote mark which caused a problem in constructing the "plain sentence" with the Python treatment and 16 because the features were not created for the sentences. In FN, there are 197 instances in the FN corpus. 41 because of faulty sentences and 156 because the preposition had not been tagged.

The problem files identify several other types. There are 4461 occurrences of the type "Can't include structure" (when it was unable to create a plain sentence) for complements (1600) or governors (2861) in

[4] http://www.clres.com/db/ske/CorpusAnalysis.html

the corpora (3939 CPA instances: 1266 for the complements and 2613 for the governors, 229 OEC instances: 115 for the complements and 114 for the governors, and 293 FN instances: 219 for the complements and 74 for the governors). There were 1029 occurrences of the type "Can't include component" (when some intervening characters made it impossible to find the component) for the preposition (6), the complement (981), or the governor (42) in the corpora (1013 CPA instances: 3 for the prepositions, 968 for the complements, and 42 for the governors; one OEC instance: one preposition, and 15 FN instances: 2 prepositions and 13 components). There were 69 occurrences of the type "Can't end structure" (when the starting position for a component was identified, but the last token did not have an ending element) for the preposition (19), the complement (15), or the governor (35) in the corpora (63 CPA instances: 15 for the preposition, 13 for the complement, and 35 for the governor; no such OEC instances; and 6 FN instances: 4 for the prepositions and 2 for the complements).

As indicated above, we print messages identifying the character location and the length of the complement and the governor for each instance. There are 80975 such messages out of 81509 instances (534 instances were degenerate). 4224 (5.2% in which One or the both of the complement and the governor could not be identified for 4224 (5.2%) instances: 322 FN instances, 240 OEC instances, 3662 CPA instances. Both were missing for 457 instances. It should be emphasized that these messages do not assess the accuracy of the complements and the governors.

4. New Available Downloadable PDEP Data

The initial downloadable PDEP data[5] consisted only of the MySQL files database tables for the sense inventory, the properties for each sense, and the tagged corpora containing all instances. This file also included three papers describing the original TPP corpora (Litkowski, 2013), the PDEP paper (Litkowski, 2014), and a paper describing the use of the paper used to create supersenses (Schneider et al., 2015).

With this paper, the downloadable file includes additional information, focusing on the vertical file used to create the English preposition corpus in Sketch Engine and the MySQL database tables. All the data described below were compressed into a WinZip file.

- Creating the English preposition corpus
  - The vertical file is created from the original Python script[6]. This script was reimplemented locally[7].
    - The basis for the script used html files in a **data** folder. There was one file for each subcorpus (cpa.html, fn.html, and oec.html) identifying each preposition.
    - The data used for creating supersenses for complements and governors (from WordNet) were obtained from GAZ files[8].
    - The vertical file was created in the **output** folder with four **vert** files, three for each of the subcorpora and one containing the full subcorpus.

[5] http://www.clres.com/elec_dictionaries.html#pdep. Note that this link now updates with the MySQL database tables. This reference also includes an additional link to download the material described in this paper.
[6] download_parses.py as described above, from Lexical Computing. Ltd.
[7] skevert1.py
[8] adjs.gaz, advs.gaz, nouns.gaz, and verbs.gaz

    - The **messages** folder contains four **msgs.txt** file, three for each of the subcorpora and one containing the full subcorpus.
    - The **problems** folder contains four **probs.txt** file, three for each of the subcorpora and one containing the full subcorpus..
- The MySQL files to add the HTML files in the data folder needed to enable the Python code to execute, how about the GAZ files), and
    - The prepdefs.sql table consists of three fields. The number of senses (preposition patterns) has increased from the original TPP inventory for two reasons. New senses have been added as a result of tagging the CPA corpus, where evidence suggested that the previous set of senses for a preposition was not adequate. Some senses were added to split an existing sense that previously had included two different classes, e.g., for 'according to', sense 1(1), with the definition 'as stated by or in', was split so that the original sense included only 'by' instances and the new sense included 'in' instances. In all, 69 new senses were added; these new senses have been given a sense number one beyond the existing set, with '(n)' as an identifier. Another source of new senses was due to senses in the Tributary class. The 24 prepositions in this class are orthographic variants of other prepositions (such as 'betwixt' for 'between'). In these cases, all senses of the base preposition were imported into the sense inventory for the variant. A total of 241 new senses were added in this manner. The total number of senses in prepdefs.sql is 1040.
    - There are presently 26 fields used to describe each sense (preposition pattern). An overview of these fields is included in the PDPE help file (http://www.clres.com/pdep.html#patterns). A more detailed description is given when a pattern is displayed, where hovering the mouse will provide a description of each field. The original set of fields in TPP may be examined at http://www.clres.com/cgi-bin/onlineTPP/find_prep.cgi.
    - The prepcorp table contains six fields: the preposition, the source (FN, OEC, CPA), the sense tag, the instance number, the 0-based location where the preposition begins, and the sentence text. There are 82,329 senstences, with 7485 in OEC, 26739 in FN, and 48105 in CPA.
- the HTML in SkEHelp contain help files available as web pages (created from PDEP.hsc).

References

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Sydney, Australia, 594-602 (http://www.aclweb.org/anthology/W-6-1670)

Ken Litkowski. 2017. *The Preposition Corpus in Sketch Engine*. Technical Report 17-01. Damascus, MD: CL Research. (http://www.clres.com/online-papers/PrepsSkE.pdf)

Ken Litkowski. 2014. Pattern Dictionary of English Prepositions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, USA, pp. 1274-83. (http://aclweb.org/anthology/P/P14/P14-1120.pdf)

Ken Litkowski. 2013. *The Preposition Project Corpora*. Technical Report 13-01. Damascus, MD: CL Research. (http://www.clres.com/online-papers/TPPCorpora.pdf)

Diana McCarthy, Adam Kilgarriff, Milos Jakubicek, and Siva Reddy. 2015. Semantic Word Sketches. *Corpus Linguistics 2015*. Lancaster University.

Nathan Schneider, Vivek Srikumar, Jena Hwang, and Martha Palmer. 2015. A Hierarchy with, of, and for Preposition Supersenses. In *Proceedings of the 9th Linguistic Annotation Workshop*. Denver, Colorado, pp. 112-123.