The Preposition Corpus in Sketch Engine

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

Abstract

Corpora from the Pattern Dictionary of English Prepositions (PDEP) provide the basis for examining the behavior of 304 English prepositions, with 1040 senses (patterns) describing in 20 fields. The PDEP corpora comprise dependency parses for 81,509 sentences in CoNLL-X format, previously used in several studies, particularly used for disambiguation modeling. We have now put these parse data into Sketch Engine (SE), using its mechanisms for further perspectives of preposition behavior. In the process, we have also annotated each parse with additional information that provides an even richer set of data to examine preposition behavior. For each sentence, Sketch Engine identifies the sense number and the direct link location to the PDEP sense description; these references can be displayed for each concordance line. Sketch Engine data for each sentence also includes the PDEP class, the subclass, and supersense tags for the preposition complements and governors (i.e., semantic relations using the WordNet lexicographer file class).

We describe in detail how the corpora were prepared for SE, involving several scripts used in PDEP to access its databases. Some of these scripts provide additional entry points into the PDEP data, particularly for the class and subclass. We describe the use of WordNet noun, verb, adjective, and adverb supersenses to tag the complements and governors, i.e., semantic word sketches for the prepositions. The resultant preposition data within SE provides a perspective different from its usual focus on the main parts of speech, so we describe the unique aspects enabled for the PDEP corpora in considerable detail. We describe several corpus query language (CQL) queries that provide useful perspectives on preposition behavior, particularly showing preposition collocations, (semantic) word sketches, preposition thesauruses, and preposition sketch differences.

1. Introduction

The Pattern Dictionary of English Prepositions (PDEP) (Litkowski, 2014; Litkowski, 2017) provides a comprehensive set of data for describing preposition behavior. This behavior is captured in individual patterns that generally correspond to senses in the Oxford Dictionary of English (Stevenson and Soames, 2003). The Preposition Project (TPP) (Litkowski and Hargraves, 2005; Litkowski, 2013) provided sentences exemplifying preposition behavior and were tagged with senses; in many cases, this tagging resulted in an expansion of the sense inventory. The tagged instances have been used as the basis for characterizing preposition behavior in individual patterns, containing up to 20 properties for each sense. PDEP provides several online routines to examine the properties of each sense. Importantly, the tagging has been used for the development of support-vector machine models for preposition disambiguation. These models have suggested shortcomings in disambiguation and has led to more detailed examination

of the importance of individual features (Litkowski, 2017). An important aspect of these studies has been the need to examine features across prepositions in the same class and subclass.

Sketch Engine[1] (SE) provides several mechanisms for examining comprehensive perspectives for the PDEP corpora. SE generally focuses on word sketches for the main parts of speech (noun, verb, adjective, and adverb), but does not usually provide methods for describing prepositions. With the help of SE staff and a combination of SE methods and various PDEP publicly scripts, we have implemented into Sketch Engine that will enable several techniques for examining preposition behavior.

In section 2, we describe the English Preposition Corpus in Sketch Engine and the procedures by which it was generated from PDEP parses and various PDEP scripts; details of the Python script used to generate the vertical file are given in the appendix. Section 3 describes the idiosyncrasies in Sketch Engine from having a corpus that focuses on prepositions. This section focuses on searching for prepositions, particularly the use of the corpus query language, describing preposition collocations, and examining semantic word sketches for preposition complements and governors. This section also shows the use of preposition word sketches via dependency relations and semantic preposition relations. This section also describes Sketch Engine preposition thesauruses and word sketch differences between pairs of prepositions. Section 4 describes how Sketch Engine can be used to enter data into PDEP patterns, characterizing the behavior of individual senses, looking at complements and governors, and examining characteristics of PDEP classes and subclasses. Section 5 describes some general observations of preposition behavior afforded by Sketch Engine, on high preposition collocates, variations in prepositional tagging from parses, and infelicities in contexts.

2. Source of Sketch Engine English Preposition Corpus

Each sentence in the TPP corpora was parsed using the Tratz parser (Tratz and Hovy, 2011), with output in the CoNLL-X format. This format consists of a line for each token in a sentence. Each token is characterized by 14 tab-separated fields, of which only six are used in the parse output for the data entered into Sketch Engine. In the CoNLL-X format, a blank line is used to separate the sentences. The fields that are present in the parse output are the token number (starting at 1 for each sentence), the token itself, the lowercased lemma for the token with an appended one-character identification of the major part of speech, a part of speech tag for the token (slightly modified from the Penn Treebank tag set), the token number of the token upon which the instant token is dependent, and the dependency relation for the token. Files in this format are called vertical files.

For PDEP, a vertical file was created for each preposition in each TPP corpus[2]. These files form the basis for the construction of the vertical files uploaded to the sketch engine. The source parse data, in the vertical file format, contains no identifier information. To provide such linkages, additional PDEP data was used from two scripts, both containing identifying numbers, one obtaining the raw sentences with the location of the focus preposition and one obtaining the locations of the complement and the governor associated with the sentence. This information was used to generate a new vertical file, with some added attributes for each sentence, in a Python script, described in Algorithm 1.

[1] http://www.sketchengine.co.uk

[2] These files are available at http://www.clres.com/db/parses/.

Algorithm 1 Sketch Engine Vertical Files

Input: List of parse files
Input: Class, subclass, and supersense dictionaries
Output: Sketch engine vertical files

1: Read list of parse files
2: For each file, **do**
3: Get dependencies for preposition
4: Get content for each dependency
5: Get sentences for corpus, preposition, sense
6: Match sentence in corpus vertical file
7: Note position for each dependency
8: Create new vertical file with annotations
9: Get positions of preposition, complement, governor
10: Get class and subclass for preposition sense
11: Annotate lemma part of speech
12: Identify supersense tag for complement, governor
13: Print vertical file for sentence

The subdirectories at the parse link provide a list of the parse files for each corpus; each subdirectory was saved as a list and used to identify the corpus name and each preposition with a parse file in that corpus. This is line 1 in Algorithm 1. Each preposition sense in PDEP has a TPP class and subclass. Since each sentence in the corpora has a sense tag, the class and subclass will be added to the sentence structure in the sketch engine files. This data is obtained from a PDEP script[3] and placed in a dictionary used during the creation of the sketch engine vertical file. This dictionary is part of line 2 of the algorithm.

Following McCarthy et al. (2014), we have used the resources of the SuperSense Tagger (SST, Ciaramita and Altun, 2006) to annotate the content words of the complements and governors in the PDEP data. These tags identify the WordNet lexicographer class, a set of 46 classes used for organizing WordNet synsets. As used in SST, the most frequent WordNet class is used as the tag. SST essentially performs a coarse word sense disambiguation. As McCarthy et al. notes, accurate WSD is not critical, and it is likely that individual errors in disambiguation will be filtered out as noise when examining the tags in the sketch engine. For this tagging, we created a dictionary (part of line 2) of the lemmas in each of four "gazetteer" files from SST (one for each major part of speech), where we appended a part of speech code to each lemma (e.g., to distinguish nouns from verbs) to facilitate lookup for a lemma-POS combination in the creation of the sketch engine vertical files.

Steps 1 and 2 of the algorithm read the list of files for a corpus and process each one in turn, creating a vertical file for the entire set of prepositions in each corpus. Each item in the list corpus is a file containing the dependency parses for each sentence that has been processing. These yields three vertical files, each of which is uploaded into the sketch engine; these files are then used as the basis for compiling and configuring the SE for the combined corpus. We describe the major steps of the algorithm in the following subsections.

[3] http://www.clres.com/db/prepclas.php

2.1. Getting the Dependencies for the Preposition

Step 3 gets the dependencies for the instances of the preposition in each corpus, using a PDEP script.[4] This script is used in PDEP to highlight in color the complements and governors of each sentence. This script is important here because it identifies (in JSON format) the sense, the instance number, and the starting positions and lengths of the complement and the governor when these are available (approximately 92 percent of the time). The instance number is most important because it provides a linking mechanism used in subsequent steps. The locations of the complement and the governor will be used in a subsequent step in identifying where to put structure tags for these elements in the new vertical file.

The list of dependencies provides the iterate for getting the content (step 4) for each instance. This first entails getting the sentences (step 5) for the specific sense of the preposition in the given corpus. This step also makes use of a PDEP script.[5] This script returns (in JSON format) the preposition, the source, the sense, the instance number, the sentence, and the 0-based position of the preposition. This script is used in PDEP to display the corpus instances for a given preposition, corpus, and sense. Here, the instance numbers between the dependencies and the sentences are linked, serving as the basis for further processing.

2.2. Matching the Sentence in the Corpus Vertical File

As indicated above, the parse files in CoNLL-X format do not identify the sentence instance numbers. All sentences in a preposition's parse file are read into tokenized structures corresponding to the fields in these files. Step 6 of the algorithm examines the "plain sentence" corresponding to the tokenized structure, i.e., the words are concatenated into a single string with no spaces. The matching process looks at the sentences from step 5, correspondingly stripped of all spaces, until it finds the one that matches the plain sentence.

When the match is found, the positions of the preposition, the complement, and the governor are annotated into the matching raw sentence (as obtained from step 5). First, the sense from the raw sentence is noted (for later use), along with the location of the preposition and its length. Next, new sentences are created for each of the three components; making use of the location and length information, structure tags are added to the raw sentence, i.e., <prep>, <compl>, and <gov>. Along with the plain sentence and preposition sense, these three tagged sentences are passed to the routine to create the vertical file entry for this sentence (step 8).

2.3. Identifying the Token Positions of the Preposition, the Complement, and the Governor

The first step in creating the vertical file entry for a sentence involves an attempt to locate the starting and ending token positions of the preposition, the complement, and the governor. Each of these uses the same routine to identify where to include the component. This routine (step 9) strips each word from the plain sentence and the tagged sentence until a tag is encountered, enabling the return of the starting and ending positions of the tag.

[4] An example is http://www.clres.com/db/getDeps.php?corp=FN&prep=about
[5] An example is http://www.clres.com/db/prepsents.php?source=FN&prep=above&sense=1(1)

2.4. **Annotating the Tokens (Class, Subclass, and Part of Speech)**

After finding the positions of the three components, we next look up the preposition and sense in the class and subclass dictionary to obtain the values to be placed in the new vertical file (step 10). Using the part of speech tags in the CoNLL-X vertical file, we append a one-letter part of speech code to each noun, verb, adjective, or adverb lemma (step 11). These are used when the new vertical file is printed.

2.5. **Printing the New Vertical File**

Step 13 prints the new vertical file, which includes all the information from the original CoNLL-X vertical files, but with various added information. The new information adds XML structures, sometimes with attributes, in creating the new files. Each preposition is contained in a **<doc>** structure; the opening tag contains a **corpus** attribute that identifies the corpus and a **preposition** attribute that identifies the preposition. Each sentence is contained in a **<s>** structure. This structure includes (1) a **sense_label** attribute containing the PDEP sense number (e.g., "1(1)"), (2) a **class** attribute containing the TPP class (e.g., "Activity"), (3) a **subc** attribute containing the TPP subclass (e.g., "Proposed"), (4) an **inst** attribute containing the instance number of the sentence in the corpus, and (5) a **sense_desc** attribute containing a link to the PDEP pattern[6] for the given sense (allowing a Sketch Engine user to examine the behavioral properties for the sense). The information in these structures will allow more targeted examination of a preposition's properties in Sketch Engine.

After the identifying information, the new vertical file contains a line for each token in the sentence, corresponding to these tokens in the original CoNLL-X vertical files. Each line is collapsed slightly from the original, containing six fields (as identified above). Additional lines are interleaved into the original lines to provide structure tags for the three main components: (1) a preposition tag (**<prep>**) surrounding the lines for the preposition (which may include more than one line for phrasal prepositions), (2) a complement tag (**<compl>**) around the line identifying the preposition complement, and (3) a governor tag (**<gov>**) around the line identifying the preposition governor.

The complement tag and the governor tag may include an **sst** (supersense tag) attribute. This tag is identified in step 12 of the algorithm, where possible. For example, a complement "withdrawing" with the part of speech VBG (a verb gerund) has the lowercased **lempos** field "withdraw-v"; accessing the supersense dictionary for this sense yields the tag "verb.motion" which is recorded as the **sst** attribute for the **compl** structure. Many complements and governors will not have **sst** attributes; e.g., personal pronouns will not have an **sst** attribute.

Running the Python script for the three corpora results in three vertical files, one for each corpus. These are uploaded to the sketch engine and compiled into a form ready for searching and other sketch engine functions. The script generates 80,369 sentences, compared to 81,509 sentences listed in PDEP. The Python script identifies problematic cases, written to separate files; these have not yet been examined in detail.

[6] An example is http://clres.com/db/TPPpattern.php?prep=on%20the%20point%20of&sense=1(1)

3. Examining the English Preposition Corpus in Sketch Engine

Since prepositions are not normally a focus of investigation in Sketch Engine, we describe additional perspectives on preposition behavior made possible with this corpus. In general, the functionality in Sketch Engine is accompanied by help pages which provide considerable detail. These are provided by means of a help icon containing a question mark; clicking on the icon opens a help page in a separate window. These help pages will supplement what is described below for the preposition corpus.

The preposition corpus is described in SE in the menu item "**Corpus info**". This shows that the corpus was generated using 571 files from three corpora and 289 prepositions. The corpus contains 80,369 sentences, 2.43 million tokens, and 2.14 million words. The corpus contains 95,272 distinct words (84,058 distinct lowercase words) and 61,330 distinct lemmas. The corpus contains 46 distinct grammatical tags and 50 distinct dependency relations (deprels). This page also has a link to a corpus description (this document) and to a description of the dependency relations (the syntactic dependency guide appendix in Tratz (2011)).

Each sentence structure (**<s>**) contains one of 14 class labels, one of 67 subclass (**subc**) labels, an instance number, the PDEP sense label, and a sense description (**sense_desc**) which provides a link to its PDEP pattern. In most of the sentences, a structure is given for the preposition (<**prep**>, in 80,363 sentences), the preposition complement (<**compl**>, 73,692 sentences), and the governor of the prepositional phrase (<**gov**>, 74,900 sentences). Where available, the **compl** and **gov** structures contains a supersense (**sst**) tag, 45 possible values for the **compl** and 46 possible values for the **gov**.

An important consideration in examining the preposition corpus is an understanding of its representativeness. The corpus is described in detail in Litkowski (2013). The whole corpus is not representative. Each of the constituent corpora must be considered on its own. The CPA corpus is the most representative, but only for its individual prepositions. For example, both *amid* and *after* have 250 instances, but these correspond to 681 and 42366 instances in the British National Corpus, respectively. The OEC corpus attempts to include 20 instances for each sense in ODE, but these are not representative for the particular sense and the full sense inventory for a single preposition cannot be considered representative. The FrameNet corpus is drawn from instances intended to illustrate particular frames and frame elements; these are frequently skewed in the number of instances for particular frames. Notwithstanding these concerns, and taking them into account while performing searches, the results from examining are likely to yield important behavioral characteristics.

There are several mechanisms in SE for examining the corpora. These include Search, Word list, Word sketch, Thesaurus, and Sketch difference. We focus on aspects that are particularly useful for examining preposition behavior.

3.1. Searching the Corpus

SE provides a form for specifying the parameters of a search to make a concordance. The bare form consists only of a text box for entering a **simple query**, as shown in Figure 1. The form can be expanded to enable a more advanced specification for **Query types** (as shown), Context, and Text types. For the most part, the **Context** will not likely to be used; the **Text types** may be used to circumscribe the concordance and will frequently be used in CQL queries.

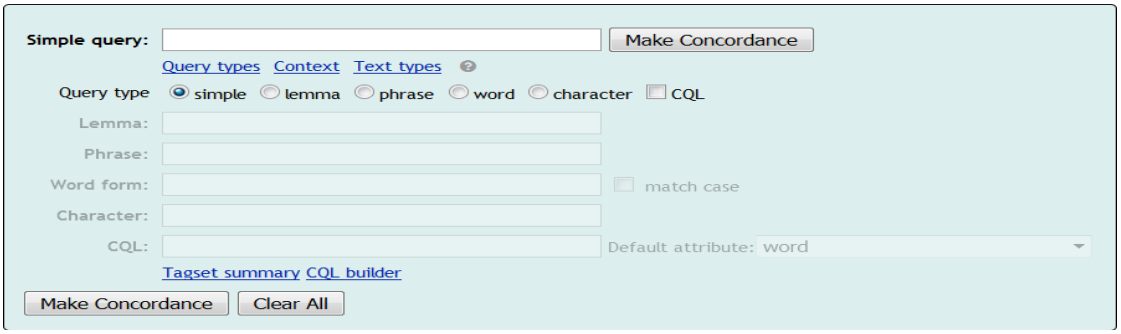

Figure 1. Query Form

When a query is performed, the result is a concordance that identifies all occurrences of a word or a phrase, as shown in Figure 2. The default displays 20 occurrences per page, each of which highlights the matches to the query. Above the instances is a simple identification of the query and the number of hits. (In parentheses, there is an identification of the number of hits per million; this is computed as the number of occurrences of the search term, divided by the number of tokens in the corpus. This denominator is used for phrases as well, i.e., not considering that the phrase consists of multiple words. As mentioned above, the number of hits should not be taken as representative.) Finally, there is a circled "i", which, when clicked, gives a detailed specification of the query parameters. The specification is linked and will bring up the query form with the specification in the CQL field.

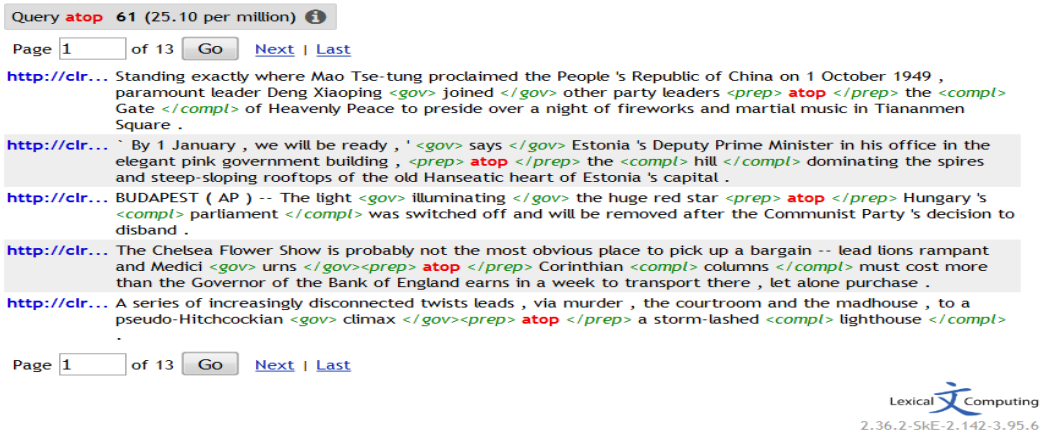

Figure 2. Search Results

The available query types are **simple**, **lemma**, **phrase**, **word**, **character**, and **CQL** (corpus query language specification). The **simple** and **phrase** queries perform essentially the same, finding all occurrences of what is entered. Since prepositions are not inflected, use of **lemma**, **word**, and **character** do not seem to provide any benefit, although they may be useful when examining other aspects of the surrounding context. When a preposition is entered as a simple query, without any further specification, the entire corpus is searched, without regard to where it occurs, frequently beyond the sentences that were identified in the preposition corpora for the specific preposition. A frequency analysis on the document corpus or preposition will show the extent to which the preposition is distributed across the corpora and the preposition files. The CQL query will be described in more detail below; this is important in characterizing the preposition queries.

When a concordance is constructed, a set of menu options is displayed. For the preposition corpus, **View options**, **Frequency**, and **Collocations** are the most relevant for characterizing preposition behavior.

The **View options** first identify how the concordance is to be displayed, either as **KWIC** (key word in context) so that the key words are aligned or as **Sentence** to display exactly the sentences for the corpus instances (as in Figure 2). Other **View options** (Figure 3) will identify what attributes, structures, and references are displayed. Usually, the **word** attribute is shown so that the concordance will display the words of a sentence; other attributes can be indicated (e.g., **lempos**, **tag**, and **deprel**), perhaps to show all tokens and/or as tooltips. If the **Structures** are selected, the affected words will be shown. For the **References** selected in Figure 3, the blue script to the left of a sentence displays the references (as in the example of Figure 2). When a concordance line is selected (to its left), a full display of the references is given, as in Figure 4. Several of these items are significant: (1) two lines identify the preposition and the corpus from which this sentence was taken; (2) a line identifies the sense number with which this sentence has been tagged; (3) a line provides a clickable link to PDEP and the PDEP pattern details for this sense; and (4) the **class** and **subc** lines identify the class and the subclass to which this sentence has been assigned.

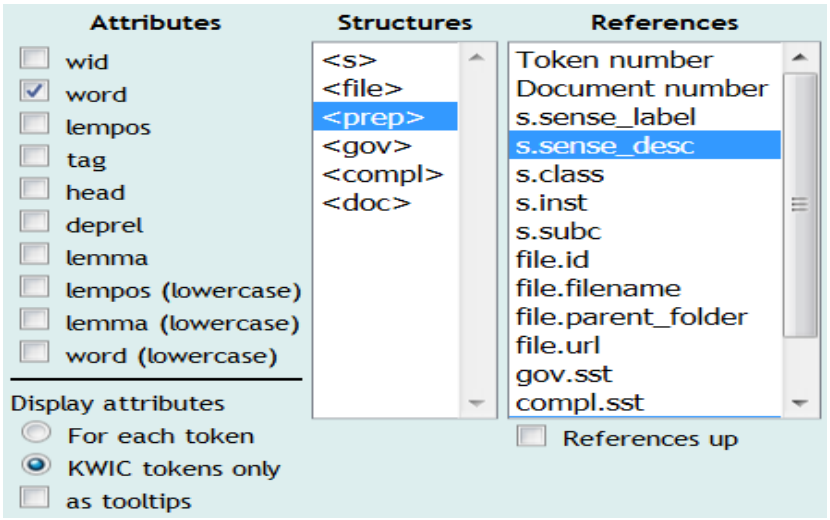

Figure 3. View Options

s.sense_label	1(1)
s.sense_desc	http://clres.com/db/TPPpattern.php?prep=atop&sense=1(1)
s.class	Spatial
s.inst	2
s.subc	Above
file.id	file4071902
file.filename	preps_cpa3.vert
doc.preposition	atop
doc.corpus	cpa

Figure 4. Concordance References

Under the **Frequency** option, the **Multilevel frequency distribution** shows selected attributes and position related to the node position and the **Text type frequency distribution** examines the distribution of the search result over various structure attributes. The sub-items for **Frequency** are **Node tags**, **Node**

forms, and **Text types**. Since the node tags and forms when the node is the preposition, the values for these are essential just frequencies, with no variation. When **Text types** is selected, all structures are shown, as in Figure 5.

	Corpus	Frequency
P | N	oec	5
P | N	cpa	2
	Sentence class	**Frequency**
P | N	Spatial	7
	Sentence subclass	**Frequency**
P | N	Behind	7
	Preposition	**Frequency**
P | N	abaft	7
	Complement supersense tag	**Frequency**
	Governor supersense tag	**Frequency**
	Sense label	**Frequency**
P | N	1(1)	7
	Sense description	**Frequency**
P | N	http://clres.com/db/TPPpattern.php?prep=abaft&sense=1(1)	7

Figure 5. Text Types Frequency for "abaft"

When the preposition has several senses, classes, subclasses, and corpus instances, Figure 5 will provide a comprehensive summary of the preposition characteristics. However, when the node specifies a preposition, the complement or governor supersense tags is not identified. For the supersense tags, the frequencies will be generated only when the search is looking at the complements or the governors (using CQL searches for these items). The resulting table has three columns: the structure attribute, the frequency, and the relative frequency. The relative frequency of the search result compares the frequency of the specific text type to the whole corpus. Since this corpus is not subdivided into the usual text types involved in this type of analysis (e.g., "spoken" versus "written"), the relative frequency is not a useful statistic. As an example, Figure 6 shows the frequency of the supersense tags occurring two or more times for the 53 complements of *atop* in the subclass "Above".

Frequency list

Frequency limit: 2 Set limit

	compl.sst	Frequency	Rel [%]
P | N	noun.artifact	18	10,048.70
P | N	noun.object	7	18,552.50
P | N	noun.group	4	3,562.20
P | N	noun.plant	3	23,881.10
P | N	noun.shape	2	34,565.60
P | N	noun.person	2	1,543.90
P | N	noun.body	2	5,560.10
P | N	noun.animal	2	14,182.10

Figure 6. Complement Supersenses for "atop"

Under the **Frequency** option, the **Multilevel frequency distribution** can be used to examine various phrasal combinations that occur in conjunction with the search result. This can include an examination of

the word, lemma, tag, or dependency relation (deprel) in up to four positions relative to the node result. We have not often used these frequency distributions.

Similar results can be examined under the **Collocations** option, which will also rank each collocation candidate with a statistic. When SE has generated a concordance, it can be examined in more detail via the **Collocations** option. This form provides options to specify the attribute, the range, the minimum frequency, the scoring functions, and the score used to sort the results for collocation candidates. The attribute can be the **word**, the **lemma**, the **tag**, the **head**, the **deprel** (dependency relation), the lowercased **word**, and the lowercased **lemma**. Of these, the **deprel** seems most informative for the preposition corpus, along with the use of the **logDice** score for sorting the results. The logDice score is considered to be the most useful,

$$logDice = 14 + log_2 \frac{2f_{xy}}{f_x + f_y} \quad (1)$$

where f_x is the number of occurrences of the keyword that gave the concordance, f_y is the number of occurrences in the collocate in the whole corpus (referred to as the candidate count for the attribute being examined), and f_{xy} is the number of occurrences of the collocate (the co-occurrence count).

This statistic measures the occurrence of a particular attribute within the search result compared to the occurrence of the attribute within the total corpus. Even though the total preposition corpus is not considered representative, it is likely that the results for the individual attributes are reasonably meaningful. (Similar results can be obtained from the Word sketch option, although it will not be able to examine collocations for phrasal prepositions.)

3.1.1.1. Query Language (CQL)

The CQL search option provides an ability to perform more detailed searches; this is important in specified the preposition instead of just using the **simple** or **phrase** query form. A description of how a CQL search is specified is available to the Sketch Engine site.[7] CQL basics provide syntax using brackets for words and attributes. For the most part, the preposition corpus will use CQL search structures, particularly focused on <**prep**>, <**compl**>, and <**gov**> structures. The following bullets identify useful CQL structures for prepositions.

- <**prep**> **[]** </**prep**>: This is the most general global search. It will create a concordance of all the instances that have been tagged in a **prep** structure. However, this tag does not surround phrasal prepositions, so it only forms the concordance for single preposition tokens (55,288 sentences).
- <**prep**> **[]+** </**prep**> **within** <**s/**>: To retrieve phrasal prepositions, i.e., having more than one or any tokens ("**[]+**" or "**[]{1,4}**") in the structure as well (80,344 sentences), but will cross sentence boundaries (which can be eliminated by specifying "**within** <**s** />").
- <**prep**> **[word="in" | word="after"] [word="the"] [word="fashion"] [word = "of"]** </**prep**> **within** <**s** />: We can enter specific words inside the brackets to retrieve just the sentences for a specific preposition; we can also enter Boolean expressions to retrieve the sentences for multiple prepositions.

[7] https://www.sketchengine.co.uk/documentation/corpus-querying/

- **<prep> []+ </prep> within <doc corp="cpa"/> within <s prep="across"/> within <s sense_label="1\(1\)" />**: We can enter a **within** statement to limit the concordance to a specific **corpus**, a specific **prep**, and a specific **sense_label** in the **<doc>** and **<s>** structures.
- **<compl sst=".*" />**: This identifies all the complement supersense tags. This concordance can then be used to examine the tags in more detail.
- **<compl> [] </compl> within <doc corp="cpa"/> within <s prep="after"/> within <s sense_label="1\(1\)" />**: The concordance will limit to the specified corpus, preposition, and sense label and will highlight the keyword in context in the preposition complement. This query search will provide the (WordNet) semantic sketches for the complement.
- **<gov sst=".*" />**: This identifies all the governor supersense tags. This concordance can then be used to examine the tags in more detail.
- **<gov> [] </gov> within <doc corp="cpa"/> within <s prep="after"/> within <s sense_label="1\(1\)" />**: As above, the concordance identifies the corpus, preposition, and sense label, but the word in context is the preposition governor.
- **<gov> [] </gov> []{2,5} <prep> [] </prep> []{2,5} <compl> [] </compl>**: This search identifies the prototypical prepositional phrase pattern. It looks for the governor, followed by the preposition and the preposition complement.

The CQL queries described above provide the basic forms that can be used in more details as well as the ability to characterize the behavior for document corpora, prepositions, sense labels, classes and subclasses, and supersense tags for complements and governors. The Text types section, described below, can assist the construction of CQL queries (see also in Figure 7).

3.1.2. Context

The **Context** option in the search query allows a specification of lemmas in the context of the search term. For the most part, this option is not likely to be used in investigating the preposition corpus. However, when a preposition is entered as a simple query, without any further specification, the entire corpus is searched, without regard to where it occurs, frequently beyond the sentences that were identified in the preposition corpora for the specific preposition. In such cases, it may be useful to examine the occurrence of the preposition in contexts beyond those sentences tagged for the specific preposition. Such cases may also be identified by performing frequency analyses for the concordances that are generated, as described above.

3.1.3. Text types

The **Text types** section in the search query provides the ability to specify portions of corpus structures to be used in generating a concordance. For the preposition corpus, there are five structures: (1) **Corpus**, (2) **Sentence**, (3) **Preposition**, (4) **Complement**, and (5) **Governor**. In general, the types included here identify the various structures in the corpus. Several of these will be relevant for the preposition concordance analyses:

- the **Corpus** structure (**<doc>**) contains the "corp" attribute, which identifies one or more of the three corpora (cpa, oec, or fn), allowing focus to a desired corpus
- the **Sentence** structure (**<s>**) contains six attributes: (1) the "prep" preposition (289 values), (2) the "class" sentence class (12 values, plus infelicitous sentences using **pv** or **x** class names), (3)

the "subc" sentence subclass (67 values), (4) the "sense_desc" sense description (1012 links that identify a PDEP pattern link to a specific preposition and sense number), (5) the "sense_label" sense label (169 sense numbers), and (6) the "inst" sentence instance number (26,205 numbers)

- the **Preposition** structure (**<prep>**) surrounds the preposition sentence (80,363)
- the **Complement** structure (**<compl>**) surrounds the 73,692 complements with 45 supersense tags (i.e., WordNet lexicographer file names)
- the **Governor** structure (**<gov>**) surrounds the 74,900 governors with 46 supersense tags (i.e., WordNet lexicographer file names)

In general, the text types limit the sentences that will be included in a concordance. However, these selections must be used carefully to produce what is desired. Usually, the CQL query type should begin with one of the **<prep>**, **<compl>**, or **<gov>** structures, containing "[]+" as the minimal. After this, further information is specified in the CQL, using one or more of the text types in the form.

The **Corpus** can be used to specify one of the three corpora. The **Preposition** is a text type that contains a drop-down list (where phrasal prepositions require "%20" for any spaces) when the user starts entered characters. The **Sentence** classes and subclasses are checkboxes for each of the possible values. The **Sentence** "sense_label" is a drop-down list of the 169 possible values to limit to specific senses for a preposition; all sense labels include parentheses and need to be preceded by backslashes. When they are specified in **Text type** form and a concordance is obtained, the resultant concordance description with the circled "i", will show the CQL specification with the text type included as in Figure 7 (which shows a **<prep>** structure with a specified class and sense_label in the sentence structure).

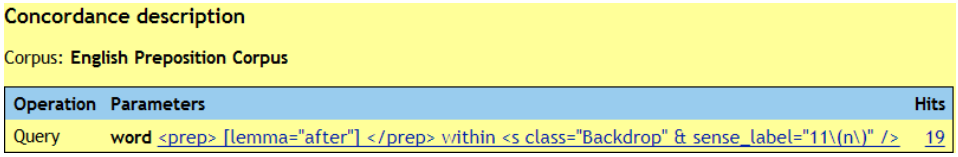

Figure 7. Concordance Description

The **Sentence** sense description can be specified in the **Text types**, and are provided in a drop-down list; these are not useful for specifying with this form (see Figures 4 and 5 for these values). The **Sentence** instance numbers are not included in the **Text types**; these are not of any meaning. The supersense tags are listed in checkboxes for the **Complement** and **Governor** structures, i.e., the various WordNet lexicographer file names. These cannot be used to obtain in this part of the form, since these checkboxes contain a period in the name (e.g., **noun.event**) which need to be escaped.

3.2. Word List

The **Word list** option lists the frequency in the preposition corpus for a specified attribute. For this corpus, one of three search attributes can be used: (1) **Positional attributes**, (2) **Word sketch**, or (3) **Text types**.

- Positional attributes include **wid** (token identifier, not used), **word**, **lempos** (lemma with part of speech), **tag**, **head** (number with head token identifier, not used), **deprel** (dependency relation), **lemma**, **lempos (lowercase)** (same as **lempos**), **lemma (lowercase)** (same as **lemma**), and **word (lowercase)**. The list for any positional attribute covers the values in the full preposition corpus.

Only the **tag** and the **deprel** are meaningful for the corpus, unless a subcorpus is created. The word and lemma options essentially duplicate frequency lists that might be found with any corpus, e.g., showing prepositions and articles as the most frequent items.

- Word sketch includes only **collocations**. The word list identifies two-word collocations, for the entire corpus, not limited to collocations involving a tagged preposition.
- Text types include **Sense label**, **Sense description**, **Sentence class**, **Sentence instance**, **Sentence subclass**, **Sentence preposition**, **Complement supersense tag**, **Governor supersense tag**, and **Corpus**. The word list for each of these items only identifies the number of tokens in sentences with the specified text type. Only the supersense tags identify the frequency of occurrences with the specified structure.

The word list also provides filter and output options. We have not used these options for any specific use for the preposition corpus.

3.3. Word Sketch

The **Word sketch** option provides a one-page summary of categorized collocations for the behavior of a lemma, via grammatical relations that have been specified in the word sketch grammar (included in the information for the preposition corpus). This is obtained by specifying a **lemma**, usually a preposition when examining the preposition behavior. Multiword phrasal prepositions cannot be entered into the word sketch lemma; this will be discussed below. Word sketch also allows various advanced options.

3.3.1. Sketch Options

The main option is the specification of the grammatical relations (gramrels) identified in the sketch grammar. Although each gramrel can be ticked in a checkbox, there are two groups: (1) CoNLL relations generated from the vertical dependency relations and (2) prepositional semantic relations from corpus attributes, identifying multiword prepositions and combinations of preposition and complement or governor. Each of these groups are discussed in detail below.

Several options specify how the gramrels are displayed. Each gramrel includes the grammatical category, the total frequency of the collocation, and the frequency of each collocation, which will display the concordance. Other options include the number of gramrel columns, the minimum frequency to show the gramrel, the maximum number of items in a gramrel, how the gramrels are sorting, and whether to show the longest-commonest match. These are described in more detail for the other options in the Sketch Engine user guide.

3.3.2. CoNLL Relations

When examining only gramrels for CoNLL dependency relations (see Figure 8 for an example), the word sketch lists lemmas that have the minimum frequency. For each lemma, the logDice score is shown (see Equation 1). In this equation, the number of occurrences of the collocate is the frequency shown in the list (e.g., “pcomp” and “which”). The number of occurrences of the keyword is the overall frequency (e.g., the combination of “from” and “pcomp”). The number of occurrences of the collocate in the whole corpus is determining by searching the **lempos_lc** for the corpus (“which-x”) and then identifying the frequency of the gramrel (e.g., “pcomp”).

The gramrels meeting the minimums (having the minimum number of lemmas) are ordered by the total frequency for the deprel. Within each gramrel, the lemmas are ordered by the logDice. The gramrel will contain all lemmas above the minimum frequency, for a maximum number specified. For lemmas having a specified frequency (e.g., 50), the lemma appears in bold and has "+" to indicate that multiword word sketch links can be examine in more detail for the lemma. In such cases, a word sketch may provide additional lemmas to form larger collocations.

While other lemmas can be examined word sketches in the preposition corpus, this will be used primarily for examining preposition collocations. As a result, the most frequent gramrel is likely to be for the deprel *pcomp*, i.e., the preposition complement. The lemmas in *pcomp* will identify the frequent collocations for the preposition. For some prepositions with few instances, the minimum frequency may require set to 1. Even for a preposition with a high frequency in the corpus, only a few gramrels (i.e., deprels) will appear in sketch; e.g., the sketch for *of* has only 9 gramrels in addition to *pcomp*.

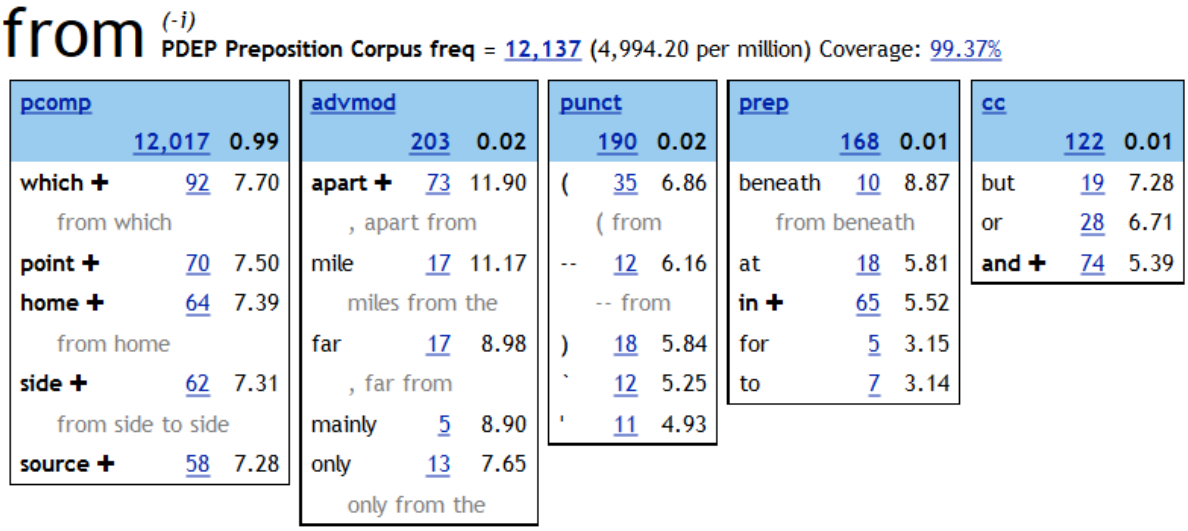

*Figure 8. Word Sketch for **from***

The word sketch is limited to a single word as the lemma (i.e., not phrasal prepositions). When the lemma is a preposition that is the first word of phrasal prepositions (such as *in* or *with*), the lemmas in the *pcomp* gramrel will correspond to multiword word sketches. When examining the concordance of such instances, the general source of the frequency will correspond to sentences with the phrasal preposition. Although the other gramrels in addition to *pcomp* are much smaller in frequency, they can be indicative of significant behavior. For example, the *advmod* in Figure 8 corresponds to the phrasal preposition *apart from*. The *punct* and *prep* gramrels also identify punctuations or additional preposition collocations for *from*.

3.3.3. **Prepositional Semantic Relations**

The word sketch for the preposition corpus includes 11 gramrels in the sketch grammar identifying multiword prepositions and combinations of preposition and complement or governor. The object of these gramrels is to characterize semantic relations for the complements and the governors. The procedures for

using these gramrels involve additional steps since the preposition corpus has several nuances to be considered.

The first procedure is designed to obtain word sketches for phrasal prepositions. This shows that the CQL query using all preposition structures indicates sentences for 289 prepositions, with 126 single-word prepositions and 163 phrasal prepositions. The single-word prepositions can be entered directly as the lemma to obtain the word sketches. For phrasal prepositions, the mechanism is to use the first word of the phrase for the lemma. When the word sketch is displayed (see Figure 9), the first gramrel is labeled as "multiword prepositions with "%w"", with the input lemma listed in the gramrel head. The lemmas listed in the gramrel constitute each of the phrasal prepositions that begin with word sketch lemma. To ensure that all phrases are shown, the minimum frequency should be set to 1 and ensure that the maximum number of items in the gramrel will show all phrases. In this gramrel, when a phrase has a "+", clicking on the phrase will generate a following word sketch for the phrase. In addition, the minimum frequency for multiword word sketch links should also be set to 1. For example, using "in" as the initiating lemma, 54 additional phrases are including; when clicking on "in_the_name_of +", the subsequent semantic relations will be shown (characterizing as "in-i filtered by in_the_name_of-x").

in *(preposition)*
English Preposition Corpus freq = 50,665 (20,847.92 per million)

multiword prepositions with "in"			words complementing preposition "in"			semantic classes complementing preposition "in"			words governing preposition "in"			semantic classes governing preposition "in"		
	7,720	15.24		2,601	5.13		2,601	5.13		1,344	2.65		1,344	2.65
in_the_name_of +	280	10.16	direction +	53	9.34	noun.location +	337	10.84	be +	93	9.96	verb.contact +	150	10.49
in_front_of +	280	10.16	way	34	8.68	noun.artifact +	407	10.45	have	11	7.64	verb.motion +	148	9.94
in_the_course_of +	270	10.11	eyes	29	8.46	noun.communication +	228	10.35	make	8	7.48	noun.person +	63	9.89
in_spite_of +	270	10.11	area	28	8.38	noun.attribute +	158	10.16	soak	6	7.18	verb.change +	67	9.85
in_excess_of +	270	10.11	voice	27	8.38	noun.cognition +	150	9.92	swim	6	7.14	verb.communication +	78	9.80

*Figure 9. Semantic Relation Word Sketch for **in***

Most of the phrasal prepositions have a single-word preposition as its first word (such as "by" or "in"). In some cases, the first word may not be a preposition, such as in "upwards of" or "a cut above". In general, using the "auto" part of speech for the lemma will still have the word sketch desired result. For "according to", it will be necessary to use "accord" as the lemma; this generates a multiword preposition for "accord_to +" as the desired sketch. Some investigation may be needed to examine the setting to obtain the word sketch.

Four gramrels are used for single-word prepositions, as in Figure 9. Two gramrels list for the complements and two gramrels list for the governors, one for words and one for semantic classes. The words list the lemmas; the semantic classes list the supersense tags (i.e., the WordNet file names). The word lemmas or semantic class supersense tags can also be generated using CQL queries; the benefit with the word sketch is that it will generate the four gramrels together, rather than requiring the four CQL queries specifying each.

To understand the logDice calculations for grammatical relations, written (*word*$_1$, *gramrel*, *word*$_2$), with

$$||w_1, R, w_2||. \quad (2)$$

For w_1, we will be specify a preposition, where we can use ".*" or the word, such as "in-i" (the lempos), or even combinations, such as "in-i| from-i". For R, we specify the grammatical relation (gramrel) where one of the names in the gramrel set. These can be a shortened name, such as "semantic classes complementing". For w_2, we will specify a "word", which can be a specific word or a semantic class.

As an example of how the logDice is computed, we use the "direction" lemma in Figure 9. Let w_1 = "in-i" (i.e., based on the word sketch with the lemma "in"), R = "words complementing preposition .*" (the gramrel selecting the word sketch grammar), and w_2 = "direction-n" (the lemma with the noun lempos "n"). We can use a CQL query to obtain the values as "[ws ("in-i", R, "direction-n")] for 53 instances with this lemma as the complement with this preposition, "[ws ("in-i", R, ".*")] for 2601 instances corresponding to all the complement lemmas, and "[ws (".*", R, "direction-n")] for 78 instances corresponding all prepositions having "direction" as the complement lemma. The general word sketch logDice score (a variant of Equation 1) is

$$logDice = 14 + log_2 \frac{2||w_i, R, w_2||}{||w_i, R, *|| + ||*, R, w_2||} = 14 + log_2 \frac{2 * 53}{(2601 + 78)} = 9.34 \qquad (3)$$

indicating in this case that "direction" in a highly collocate with "in" compared to other prepositions. As indicated, the first two components of the logDice computation are shown directly in the grammatical relation, while the other component is not intuitive and needs to be identified from the CQL query word sketch.

As suggested in Figure 9, the words and particularly the semantic classes provide interesting behavior descriptions, indicating the most frequent items. In addition, more detail about semantic classes can be obtained by following the multiword sketches (i.e., those with "+"). For example, clicking on **noun.location** for "in", the multiword sketch provides details for "in-i filtered by noun.location-x", in Figure 10. We can see contrasts between the multiword gramrels with the gramrels of the base. For complementing words, there is a difference between Figure 9 and Figure 10; the word **direction** is still the same top word at the same frequency, but the logDice increases from 9.34 to 12.03. For the governing words, few words meet the minimum value and the logDice is much lower than for the full table. For the governing semantic classes, none of the high classes appear in the multiword sketch. Thus, the multiword sketches provide a difference from the full word sketch.

in ... area *(preposition)* Alternative PoS: noun (50) adverb (49) adjective (30)
English Preposition Corpus freq = 337 (138.67 per million)
(in-i filtered by noun.location-x)

in: words complementing preposition "in"		
	337	100.00
direction +	53	12.03
area	28	10.93
london	13	10.01
place	11	9.78
england	9	9.59

in: words governing preposition "in"		
	184	54.60
be	10	7.43

in: semantic classes governing preposition "in"		
	184	54.60
noun.object	12	9.99
noun.group	8	8.46
noun.act	16	8.44
verb.motion	36	8.30
noun.person	9	8.18

*Figure 10. Detailed Word Sketch for Semantic Class for **in***

Four gramrels for the single-word preposition (not shown, but below from those in Figure 9) provide the summaries of word and semantic classes for all those from the multiword prepositions beginning phrasal prepositions with the word sketch lemma (e.g., the 54 multiword prepositions for "in"). In general, it is unlikely that there is any semantic thread from these multiword. As also suggested above, it is probably desired to follow the multiword sketch for the phrasal preposition to understand their behavior.

The final two gramrels that may be generated from the word sketch arise when a complement is a preposition or a governor is a preposition and these positions have another preposition structure. For example, "in" occurs in some sentences as the governor and there is another preposition that occurs as a

focus of a **prep** structure (such as "preparatory to", "as well as", and "ahead of"). These types of gramrels seldom occur; it is not clear that such behavior is of considerable interest.

3.4. Thesaurus

The **Thesaurus** option provides a form that requires a single-word preposition lemma (i.e., phrasal prepositions are not included). For the preposition corpus, the results generate a list of similar prepositions, as shown in Figure 11. This list is ordered by an association score. The list also shows the frequency of each preposition in the corpus. Next to the list is a word cloud, with the most similar prepositions in the largest size. Each preposition may be selected (either from the list or the cloud) to show the sketch difference from the seed lemma. The form has a set of advanced options: (1) a maximum number of items, (2) a minimum score, (3) whether to show the head word in the cloud, (4) whether to cluster items, and (5) the minimum similarity between cluster items (a higher number produces smaller groups of words which are closer in meaning).

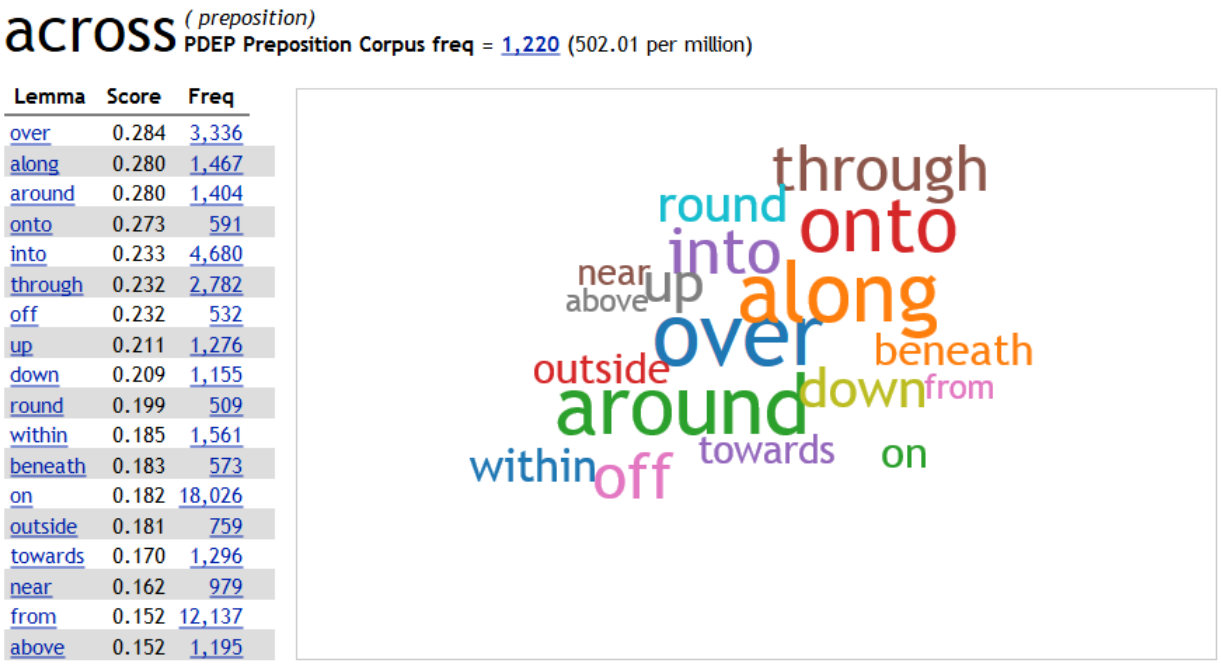

*Figure 11. Thesaurus Results for Preposition **across***

The thesaurus finds words that tend to occur in similar contexts as the target word. In general, the similarity score between two words w_1 and w_2 is computed by first determining all the overlaps where the two words share a collocation in the same grammatical relation where an association score is greater than 0. The set of all word sketch triples (*headword*, *relation*, *collocation*) is used to find applicable contexts ($ctx(w_1)$), i.e., the set of (*relation*, *collocation*) in the word sketch triples set. The logDice is used as an association score. A distance score is then computed.

The distance between two prepositions is preserved, i.e., when changing the seed word back and forth. However, the position in the list will change. For example, for *at*, the closest word is *from* (with a score of 0.410), whereas with *from*, *at* is in the 9th position. (Since the matrix is symmetric, one could conceivably prepare a matrix with the prepositions as rows and columns. Then, one could highlight the cells that show

the highest scores. I don't know how such a matrix could be generated, other than going through the prepositions one by one and saving the results.)

When the minimum similarity between cluster items is increased, fewer prepositions are included in the clusters and there seems to be a tendency toward greater similarity. In addition, it almost seems as if the clusters hold more generally, and not just in relation to the seed lemma. But also, the clusters may not meet intuitive expectations (e.g., *under* doesn't show similarity with *underneath* or *below*).

3.5. **Sketch Difference**

The **Sketch diff** option provides a form that allows the comparison of two lemmas. (This option can also be accessed from Thesaurus results.) This option shows the differences in the word sketches between two lemmas, as described in Figure 12. The basic form asks for the first and the second lemma. There are several advanced options.

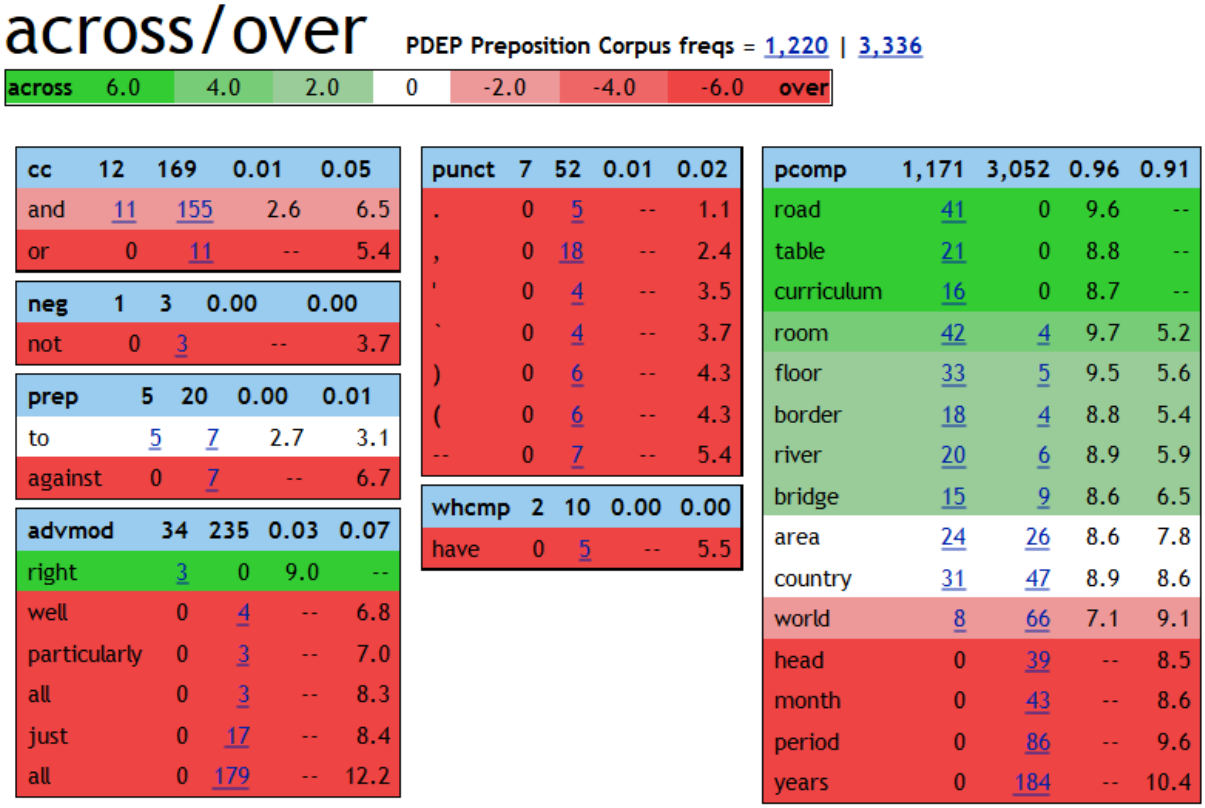

*Figure 12. Sketch Difference between **across** and **over***

The result assigns red and green colors to the two lemmas. The collocates in each color tend to combine with the lemma of the same color, with white collocates tending to combine with both lemmas. Bolder shades indicate stronger collocations. The top line shows the two lemmas and their respective frequencies within the corpus. The next line shows the two lemmas at the ends of a spectrum of color, with the color boldness corresponding to the collocation strength. The results are shown in boxes, each of which is labeled by the grammatical relation, i.e., those identifying in the word sketch grammar (if not clear,

looking at examples will help identify the category). For the preposition corpus, the grammatical categories are the dependency relations (deprels) and the prepositional semantic relations.

The boxes show the results in the gradations of color, from green to red (listed as +6.0 to -6.0, with no explanation). A relation is shown when a lemma or a semantic class has the minimum frequency threshold. Each box has a line showing the relation with the total frequencies and the logDice scores for each preposition. Within each box, the lemma or the semantic class is given on each line, along with the frequencies and logDice scores of the relation for each preposition (clickable so that these instances can be examined, with the lemma highlighted). The frequencies may not add to the total for the gramrel, since the other items do not attain the minimum frequency threshold or the maximum number of items that can appear in a block, as specified in advanced options.

With the advanced options, one can specify that the differences be shown in a single box for each relation, or in common/exclusive blocks. With a single box, one can see a gradation from one word to the other, with common lemmas in white in between. This perhaps allows a little bit of judgment as to the overlap between the two lemmas.

4. Use of Sketch Engine for PDEP Database

The primary purpose of uploading the PDEP corpora in Sketch Engine is to provide a systematic basis for completing fields in the patterns describing each sense of a preposition. Specifically, this includes a syntactic and a semantic characterization of the complement and the governor. In the initial creation of the TPP database, the complement and the governor were described using general lexicographic principles. In addition, TPP also identified a syntactic position (e.g., noun postmodifier, adjunct, verb complement, or adjective complement). These can be made more precise by examining the sketch engine data. For example, we would begin with a specification such as **<compl> [tag="N.*"] </compl> within <s prep="after" & sense_label="7\(3\)" /> within <doc corp="fn" />** and examine frequencies of characteristics of such a concordance.

Another major use afforded by the sketch engine data is the ability to examine the PDEP classes and subclasses. Srikumar and Roth (2013) showed the utility of examining preposition behavior across prepositions. Litkowski (2017) performed a series of feature analyses across prepositions using Kullback-Leibler and Jensen-Shannon divergences, based on the PDEP classes. With the addition of PDEP subclasses, it may be possible to judge the consistency and differences among subclasses compared to their classes. The TPP data also identified substitutable prepositions, viewed as somewhat narrower than classes or subclasses.

We describe procedures for carrying out these analyses. In formulating the queries for these procedures, it is useful to keep in mind that they can be performed over the entire corpus (remembering that it is not representative) or a specific subcorpus (CPA, OEC, or FrameNet). When these analyses are performed, it may be useful to save the results.

4.1. Examining the Complement or Governor of Preposition Senses

To examine the complements or the governors, it is necessary to begin the CQL specification with either **<compl> [] </compl>** or **<gov> [] </gov>**. These bare specifications identify all 73692 complements or 74900 governors. We use the text type fields of a search query to narrow the concordances. For the corpus

structure or the sentence structure, the text type can also be specified by **within** clauses. For specific complement or supersense tags, the CQL needs to be identified inside the bracket. The general PDEP pattern would specify a preposition and a sense, possibly looking within one subcorpus at a time. Following Srikumar and Roth, instead of a single preposition and sense, we can use a single subclass, several subclasses, or a full class; using this form will make it possible to include several preposition-senses together.

The most efficient procedures for obtaining the characteristics of the complements and the governors use **Frequency** (compute frequencies) in the left menu. Under the **Multiple frequency distribution**, useful results suffice the first level and look at the **Node** for the various attributes, selecting from their drop-down list. The **deprel** and **tag** attributes will provide an initial overview, each of which have fewer types. For **tag**, most types are "N.*" or "V.*"; to obtain summaries of these types, more detailed concordances will be necessary. In some cases, **word**, **lemma**, and **lempos (lowercase)** may be useful, although these attributes generally have more many items. The semantic characteristics of the complement or governor (i.e., supersense tags) can be seen using the **Text type frequency distribution** by selecting the category of **Complement supersense tag** or **Governor supersense tag**. This will list the WordNet lexicographer file names in descending frequency. When viewing these results, it should be kept in mind that many lexical items are not assigned a supersense tag, particularly when the complement is a personal pronoun or the governor is a copular verb. When these results are saved, the number of instances without a supersense tag will also be listed.

For the complement, the most frequent dependency relation (deprel) is likely to be *pcomp* (i.e., a preposition complement), regardless of subcorpus, preposition, or sense. In general, the tag is likely to be some variant of "N.*" (i.e., some type of common or proper noun), where some prepositions or senses may be more frequent for plural nouns or proper nouns. Some prepositions or senses may have more frequent for personal pronouns, determiners, or gerunds. The supersense tags for complements will usually have *noun.** types. In general, complements will not have frequent words or lemmas or other variations, except for prepositions that have many personal pronouns or determiners. When examining the frequencies for an attribute, we might look at some oddity in the parsing, pursuing for further analysis.

For the governor, there is considerable variation in the attributes (deprels, tags, lemmas, and supersenses) among the prepositions and their senses. For example, for "after", the most frequent deprels are *ROOT*, *vch* (a verb chain), *ccomp* (complement), with most being variants of "V.*" (i.e., some type of verb) and with *verb.** supersenses. For "of", the most frequent deprels are *pcomp* (i.e., following another preposition complement) *dobj* (direct object), and *subj* (subject), mostly nouns forms ("N.*") and *noun.** supersenses. For "because of", the emphasized deprels are *vch*, *ROOT*, and *ccomp*, with a high frequent of *be-v* lowercased lemmas, and mostly types of *verb.** supersenses. These variations suggest that distributional analyses may be fruitful.

4.2. Class and Subclass Analyses

PDEP has assigned classes and subclasses to each sense. Sketch Engine provides an opportunity to question these assignments and to examine the consistency of the classes and subclasses. In general, these analyses involve a text type restriction in the **<s>** structure to specific classes or subclasses. In CQL, such a restriction is specified as **class = "value"** or **subc = "value"**. The 12 class values and the 67 subclass values are available as checkboxes under the text types. When using a CQL specification for a

preposition, complement, or governor, we can add a further specification using **within** an **<s>** containing a class or subclass. Values can also be combined in a specification, by enclosing optional values in parentheses separated by "|", e.g., **(subc = "Above" | subc = "Below")**. These values can enable unusual combinations that can examined for other properties.

There are several dimensions of properties that can be examined under the class analyses. Of primary interest are the complement and the governor properties. To access these, either **<compl> [] </compl>** or **<gov> [] </gov>** should be used as the principal specification in the CQL statement. Once a concordance has been generated, the properties can be examined in detail using the frequency and collocation tools.

The **tag** and **deprel** frequencies provide an initial overall picture. For the complements, we should expect that noun tags and **pcomp** deprels (preposition complements) will predominate. For some prepositions, wh-clauses or gerundial forms will show heightened frequencies. For the governors, we can expect verb forms and **ROOT** deprels, although in many cases the preposition phrase will be governed by a **pcomp**, suggesting that it is modifying a noun.

The supersense tags may provide the best characterization of the complements and the governors. When examining a class or subclass, it may be useful to first bring up a frequency analysis by the document preposition. Then, the sketch engine provides the ability to generate a concordance for each preposition and then to generate frequency analyses for each in turn, enabling a side-by-side comparison of the frequencies. These results can also be saved and use for more detailed divergence analyses.

5. General Observations About Preposition Behavior Using Sketch Engine

In addition to the more specific results described above, examination of the corpora in Sketch Engine provides some further insights into preposition behavior.

5.1. Collocates of Prepositions

The CQL search **<prep> []+ </prep> within <s/>** generates a concordance of all the prepositions within the English preposition corpus sentences. When examining the collocation candidates using the **deprel** attribute within five tokens to the left and to the right, **ROOT** has the highest logDice score, following by **subj**, **pcomp**, **det**, **punct**, and **amod**. This can be interpreted as saying that a prepositional phrase usually modifies the root of the sentence or its subject, it generally consists of a preposition complement modified by a determiner or an adjective, and it frequently ends some sort of punctuation.

5.2. Variations in Preposition Tagging

Sketch Engine can be used to identify errors in recognizing prepositions using the Tratz parser. When the TPP corpora were prepared, the target preposition was specified using its character position within the sentence, prior to parsing. In general, we would expect the parser to yield a **prep** deprel and an IN tag for the preposition. However, this is not the case. To investigate such cases, we use the CQL query as indicated in the previous section to examine the attributes for the single-word prepositions and the multiple words of phrasal prepositions. Using the multilevel frequency distribution option from the **Frequency** menu, we specify the deprel attribute or the tag attribute for the node position. Thus, in addition to identifying the constituents of multiple word phrasal prepositions, these attributes also identify where single-word prepositions have differences that are not expected.

For the most part, single-word and phrasal prepositions follow the expected deprel (*prep*, *prep pcomp prep*, and *prep det pcomp prep*) and tag (*IN*, *IN NN IN*, and *IN DT NN IN*) patterns, but only slightly above 72 percent of these instances. There are 399 deprel different configurations and 97 tag different configurations. In addition, there are several other attribute values for single-word prepositions, with about 40 different deprels and 25 different tags. With Sketch Engine, we can use the frequency lists to examine the deprels or tags that have given rise to unexpected values and to examine the concordances for the different configurations.

Several single-word prepositions, such as "about", are frequently confused in parsing as adverbs or particles. For phrasal prepositions, such as "apart from" or "because of", the parsing goal is to label them with the deprel "prep combo". Many of the attribute configurations may have arisen from incorrect parses or problematic spaces in attempting a sentence. Sketch Engine frequency analyses permit us to see where this goal has not been achieved.

As mentioned earlier, the FrameNet is not representative. One indication of this is that when we examine the frequency deprel and tag patterns for the FrameNet subcorpus, there are not multiple-word phrasal prepositions. That is, these sentences have only a single deprel or a single tag. However, even in these instances, about 5 percent of the instances have attributes that are not the expected *prep* or *IN* attributes and so can still be used to help in improving parsing or tokenizing techniques.

5.3. **Infelicities in Tagging Preposition Contexts**

Sketch Engine also facilitates an examination of the contexts of the preposition use to identify problematic cases. This can be done using the collocation or the frequency options for a concordance, using **View options** to display attributes as tooltips.

The collocation option is best used in conjunction with **<prep> []+ </prep>** queries. For the most part, these collocations are best used to examine 1 to 3 positions to the right of the preposition. Examination of the left context is generally not as fruitful. In general, we would expect **pcomp** to be the highest ranked deprel (corresponding to the preposition complement), with higher logDice values for **det** (determiner), **amod** (adjective modifiers), **nn** (noun modifiers), and **poss** (for possessive modifiers). All other deprels will also be listed in the collocation candidates, in decreasing logDice levels. We can be examined the concordances with each deprel collocation candidate; in addition to the preposition, the concordance will highlight the token for the deprel including with the candidate This makes it easier to determine if the expectations make sense.

The frequency option is best used to examine the complements (with **<compl> [] </compl>**) and the governors (with **<gov> [] </gov>**), specifying particular prepositions, classes, or subclasses, as described above. In general, the focus of these frequency analyses is to identify predominant syntactic or supersense tags, as described earlier. In addition, we can examine the less frequent values. The tag frequencies may reveal parsing problems or may uncover less frequent, but important possible values for the complement or the governor. The supersense frequencies may identify problems with using the most frequent WordNet sense; this may suggest that some of the more frequent supersenses may have an even higher frequency.

References

Massimiliano Ciaramita and Yasemin Altum. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. *EMNLP-'06 Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, ACL, 594-604.

Ken Litkowski. 2013. *The Preposition Project Corpora*. Technical Report 13-01. Damascus, MD: CL Research.

Ken Litkowski. 2014. Pattern Dictionary of English Prepositions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, ACL, 1274-83.

Ken Litkowski. 2017. Pattern Dictionary of English Prepositions. In M. Diab, A. Villavicencio, M. Apidianaki, V. Kordoni, A. Korhonen, P. Nakov, and M. Stevenson, editors *Essays in Lexical Semantics and Computational Lexicography – In Honor of Adam Kilgarriff*. Springer Series Text, Speech, and Language Technology. Springer.

Ken Litkowski and Orin Hargraves. 2005. The preposition project. *ACL-SIGSEM Workshop on "The Linguistic Dimensions of Prepositions and Their Use in Computational Linguistic Formalisms and Applications"*, pages 171–179.

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

Vivek Srikumar and Dan Roth. 2013. Modeling Semantic Relations Expressed by Prepositions. *Transactions of the Association for Computational Linguistics*, 1.

Angus Stevenson and Catherine Soanes (Eds.). 2003. *The Oxford Dictionary of English*. Oxford: Clarendon Press.

Stephen Tratz. 2011. *Semantically-Enriched Parsing for Natural Language Understanding*. PhD Thesis, University of Southern California.

Stephen Tratz and Eduard Hovy. 2011. A Fast, Accurate, Non-Projective, Semantically-Enriched Parser. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.

Appendix

Python Script to Create a Vertical File

This section describes the Python script (**download_parses.py**) used to create the vertical file **preposition.vert** and the file **preposition.conll** that were uploaded to the sketch engine. This script is accompanied by a **Makefile** that automates this creation, iterating over the files in the **/data** subdirectory (**(cpa|fn|oec).html**), each of which contains a list of ***.parse** files in the directory **/db/parses/(cpa|fn|oec)** at the PDEP web site. The Python script is called with one of the HTML files as argument. The output of each processing is appended to **preposition.vert**, at the completion of which, this file is used to create **preposition.conll**.

1. main

This function creates an argument parser to read the input file name (the HTML file). It defines the PDEP **domain** as http://www.clres.com/db/parses/ and **corpus_name** as a result of matching **(.*).html** from the file name. We then iterate over each **line** in the HTML file, with a regular expression search that obtains the **prep_file_name** and the **prep** from any line containing **(.*).parse**. When we have a **prep_file_name**, we get the JSON file at **r** from http://www.clres.com/db/getDeps.php with **corpus_name** and **prep** as arguments; this file contains the complement and the governor locations and lengths for all instances of the preposition in the corpus. We set **remote_file** to **r.data** and **compl_gover** (complement and governor) to the result of **json.loads** of **remote_file**. We print the opening tag for this preposition, **<doc corpus="corpus_name" preposition="prep_file_name">**. The function **get_content** is then called with the url, **corpus_name**, **prep**, and **compl_gover** as arguments (1) to link the dependency file with the preposition sentence file, (2) to link these with the sentences in the dependency parses file, and (3) to create a new vertical file with tags (for the preposition, complement, and governor) surrounded by a sentence element with a sense label and a link to a sense description and with part-of-speech tags appended to the lemmas. Finally, we print the closing tag, **</doc>**.

1.1. get_content (url, corpus_name, prep_file_name, compl_gover)

This function has the arguments **url** (the url of **CoNLL** formatted verticals for the given corpus and preposition word, e.g., http://www.clres.com/db/parses//fn/above.parse), **corpus_name** (the name of the corpus), **prep_file_name** (name of file that stands for preposition we want to investigate), and **compl_gover** (a list of JSON objects that consists of the sense, the sentence number, the complement location and its length, and the governor location and its length).

The function first gets the vertical file at the **url** and sets **prep_sents** to the result of calling **get_sentences** to get all the sentences for the specified preposition in the specified corpus in CoNLL tokenized format. The function next iterates over the elements in **compl_gover**, using **j_obj** as the iterate, with a message identifying which record is being processed, where each iterate identifies a sense, an instance number, and the locations and lengths of the complement and the governor. The function next calls the PDEP script http://www.clres.com/db/prepsents.php?source=FN&prep=about&sense=1(1) to get all the sentences that have been tagged in the specified corpus for the preposition with the given sense. Next, we iterate over these sentences, searching for the one whose 'inst' element is equal to the 'inst' element of **j_obj**. Once this link is made, we next need to find this sentence in the vertical file. This is done by iterating over the

sentences in **prep_sents**, with the iterate **sent**, calling **sent_plain** with **sent** until its result matches the string formed by concatenating the word associated with the 'sentence' element of the matched instance.

When a match with the file in the vertical file is made, **matching_sent** is set to **sent** and the variables **sense**, **p_sent**, **prep_loc**, and **prep_len** are set to the 'sense', 'sentence', 'preploc' (preposition location), and 'prep' (preposition) elements of the sentence that was matched with the 'inst' element. Next, tags are added to **p_sent**, first to surround the preposition of interest with the **<prep>** tag, and then to try to do this as well for the governor (**<gov>**) and the complement (**<compl>**), into the variables **preposition_sent**, **governor_sent**, and **complement_sent**.

If **matching_sent**, **sense**, and **preposition_sent** have values, **create_new_vert** is called with **matching_sent**, **sense**, **prep_file_name**, and the collapsed sentences **preposition_sent**, **governor_sent**, and **complement_sent** (each of which will now have a tag) to print a vertical file entry for **matching_sent**, surround by a sentence tag with attributes for the sense and a link to the PDEP pattern for that sense. Otherwise, a message is written to stderr that the sentence wasn't found for this **j_obj**.

1.2. create_new_vert (matching_sent, governor_sent, preposition_sent, complement_sent, sense, preposition)

This function prints out a new vertical file for the argument **matching_sent**. The function first calls **include_component** to identify the locations for the **<gov>**, **<prep>**, and **<compl>** tags, i.e., the beginning and ending token numbers. The function prints the opening tag for the sentence, **<s sense_lab="sense" sense_desc=http://clres.com/db/TPPpattern.php?prep=preposition&sense=sense>**. The function then iterates over the tokens in **matching_sent** setting **i** to obtain the token. If **i** is the starting or ending position for the governor, preposition, or complement, the corresponding tag is printed as part of the vertical file. The function next prints a line for the token, with tab-separators, including the token id, the token itself, the lemma (lempos), the (part-of-speech) tag, the dependency id, and the dependency relation. After printing all tokens, a closing **<s>** tag is printed.

1.3. include_component (tok_sent, plain_sent, structure)

This function marks the beginning and ending locations for the **<gov>**, **<prep>**, and **<compl>** tags in the vertical file, i.e. the **structure** argument. If **plain_sent** is empty, returns -1, -1. Otherwise, enters a while loop over the tokens in **tok_sent**, setting **token** to each token in turn. If **token.word** begins **plain_sent**, resets **plain_sent** by this word, increments a token counter, and continues to the next word. If instead, **plain_sent** begins with **<structure>**, removes the tag from **plain_sent** and sets **start** to **token.id**. If **plain_sent** begins with **</structure>**, removes the end tag from **plain_sent** and sets **end** to **token.id**. Proceeds over all tokens of the sentence, possibly unnecessarily. After the loop, if 1000 tokens have been counter, prints an error message. If both **start** and **end** have been set, returns them. Otherwise, prints an error message.

1.4. get_sentences (memory_file)

This function builds an array at **result** by iterating over **memory_file** as long as a sentence **s** is returned, appending each sentence to **result**. This array is the set of all sentences for a particular corpus and preposition, with each tokenized in CoNLL format for printing to the vertical file.

1.5. **get_sentence (memory_file)**

This function gets a sentence from **memory_file**. Starts with an empty array at **sentence**. For each **line** in **memory_file** that is not empty, splits into **spl** based on tab separators. Sets **id**, **word**, **lemma**, **tag**, **depid**, and **dep_lable** to **spl[0]**, **spl[1]**, **spl[2]**, **spl[4]**, **spl[6]**, and **spl[7]**, respectively, with an error message if there is an index error. If successful, sets **token** to the result of creating a new **Token** with these variables as argument and appends this token to **sentence**. If a line is empty, the function yields **sentence** (i.e., returns this result to the calling function **get_sentences**) and empties it for the next sentence.

1.6. **sent_plain (sentence)**

This function builds an array of the **token.word** elements of **sentence**. It then returns a string concatenating the elements of the array into a string corresponding to the sentence without any whitespace. The output of this function is used in matching the sentence with an instance number in **get_content** and in identifying the location of the preposition, the complement, and the governor tags in **include_component**.

1.7. **getsups ()**

This function builds global arrays **dict** and **subcs** from the preposition classes JSON data.[8] Each item of the JSON data has a combination **tup** (a tuple) for the 'prep' and 'sense'. This tuple is entered to the array **dict[tup]** for the 'sup' and to **subcs[tup]** for the 'subc'. These will be used in create_new_vert.

1.8. **getdicts ()**

This function builds a global array **sstags**, the dictionaries from Ciaramita and Altum (2006), containing the {nouns|verbs|adjs|advs}.gaz files. Each line is read from each line, splitting into tabs, and taking the first two as the lemma and the lexicographer file name. This array will be used to identify a complement and a governor lemma of the appropriate part of speech. This will be used in create_new_vert.

1.9. **Token (id**, **word**, **lemma**, **tag**, **depid**, **dep_lable)**

This class forms an instance of **Token** using its arguments to set the respective elements of the instance. The **id** member is set directly. All other members are set to the result of decoding the string according to *utf-8*. The **word** member may be modified if it is a double backquote, a double singlequote, or contains an ampersand. The same is done for the **lemma**. The **tag** member is set directly. The **depid** and **dep_lable** members are set directly.

The **lempos** member is set based on a call to the member function **make_lempos** with **lemma** and **tag** as arguments. If **tag** is in u"CC", u"IN", u"JJ", u"JJR", u"JJS", u"NN", u"NNS", u"NNP", u"NNPS", u"VB", u"VBD", u"VBG", u"VBN", u"VBP", u"VBZ", the first character of the tag is appended to the lemma. If **tag** is one of "RB", "RBR", "RBS", an **a** is appended. If **tag** is none of these, **x** is appended.

[8] http://www.clres.com/db/prepclas.php