WAFFLE: A Graph for WordNet Applied to Free-Form Linguistic Exploration

Berk Ekmekci & Blake Howald
Thomson Reuters Special Services, LLC
1410 Spring Hill Road, Suite 125
Mclean, VA 221022
[berk.ekmekci, blake.howald]@trssllc.com

Abstract

The WordNet database of English (Fellbaum, 1998) is a key source of semantic information for research and development of natural language processing applications. As the sophistication of these applications increases with the use of large datasets, deep learning, and graph-based methods, so should the use of WordNet. To this end, we introduce WAFFLE: WordNet Applied to FreeForm Linguistic Exploration which makes WordNet available in an open source graph data structure. The WAFFLE graph relies on platform-agnostic formats for robust interrogation and flexibility. Where existing implementations of WordNet offer dictionary-like lookup, single-degree neighborhood operations, and path-based similarity-scoring, the WAFFLE graph makes all nodes (semantic relation sets) and relationships queryable at scale, enabling local and global analysis of all relationships without the need for custom code. We demonstrate WAFFLE's ease of use, visualization capabilities, and scalable efficiency with common queries, operations, and interactions. WAFFLE is available at github.com/TRSS-NLP/WAFFLE.

1 Introduction

WordNet (Miller, 1995; Fellbaum, 1998) is a database of English words with associated lexical properties and semantic relations. For example, WordNet includes seven semantically distinct *senses* for the noun "establishment":

[establishment.n.01/constitution.n.02] - *the act of forming or establishing something*

[establishment.n.02/institution.n.01] - *an organization founded and united for a specific purpose*

[establishment.n.03/administration.n.02] - *the persons who make up a body for the purpose of administering something*

[establishment.n.04] - *a public or private structure including buildings and equipment for business or residence*

[establishment.n.05] - *any large corporation*

[establishment.n.06] - *(ecology) the process by which a plant or animal becomes established in a new habitat*

[establishment.n.07] - *the cognitive process of establishing a valid proof*

Each of these senses are organized by individual *synsets* (synonym sets) and labeled for reference with a word.part-of-speech.number structure. Sysnsets include definitions, examples, lemmas, synonyms (e.g. establishment.n.01 is equivalent to constitution.n.02) and are organized into larger hierarchical relationships (Figure 1), which can facilitate the computation of paths between synsets to quantitatively approximate word similarity. For example, there are 2 hops (steps up or down the hierarchy) between establishment.n.02 and .05 compared to 9 hops between .02 and .06 (organized institutions being more like corporations rather than plants or animals establishing a new habitat).

Figure 1 is based on noun hypernym and hyponym relations, but WordNet includes additional parts of speech (verb, adjective, adverb) and associated relations - e.g. *entailment* between verbs, *antonyms* between adjectives, and *derivationally related forms* for all parts-of-speech. WordNet has been used for building dictionary and thesaurus applications as well as a range of natural language processing tasks such as: word sense disambiguation tasks (Patwardhan et al., 2003; Navigli, 2009; Loureiro and Jorge, 2019), document retrieval (Rada et al., 1989; Srihari et al., 2000), information extraction (Stevenson and Greenwood, 2005; Atkinson et al., 2009), and querying (Bulskov et al., 2002; Li et al., 2003) for recommender (Blanco-Fernández et al., 2008) and question-answer (Tapeh

Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS), pages 147–157
Virtual Conference, November 19, 2020.

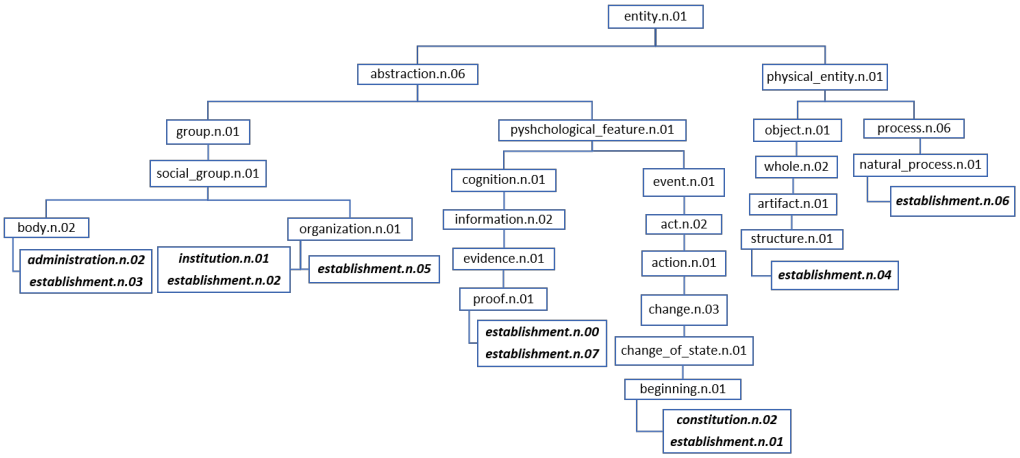

Figure 1: WordNet is-a (noun-based hypernym/hyponyms) hierarchy for *establishment*.

and Rahgozar, 2008) systems.

The current version of WordNet (117,000 synsets in version 3.1 with 27 relation types) is available through an interactive browser, APIs, and stand alone database files which can be customized.[1] However, beyond recreating common functionality, we believe there is an increasing need for the availability of WordNet in an open source graph-based data structure to support large-scale use and research (e.g. for deep learning (Yuan et al., 2016; Diao et al., 2018; Vial et al., 2019; Kobylinski and Wasiluk, 2019), hierarchical embeddings (Bernardy and Maskharashvili, 2019), and graph-based approaches generally (Naskręt et al., 2018; Pinter and Eisenstein, 2018). These use cases leverage not only the content of WordNet, but need to do so with increasing sensitivity to the *structure* of WordNet. This is not only to operate more efficiently, but to open up additional potential avenues of research. To satisfy this need, we present WAFFLE: WordNet Applied to FreeForm Linguistic Exploration as a fully-connected queryable graph representation of WordNet to provide: (1) flexibility in exploring *all* of WordNet's relations across synsets and hierarchies rather than particular part-of-speech-based subgraphs; (2) scalable processing for large datasets; and (3) support for all common operations on WordNet (look-up, similarity measures).

The remainder of this paper is structured as follows: Section 2 introduces the details of WAFFLE's graph structure, computation and descriptive statistics. Section 3 demonstrates common WordNet operations compared to non-graph structure approaches. Section 4 discusses related methods of WordNet access. Section 5 concludes with WAFFLE's access and licensing details with plans for future versions.

2 WAFFLE Graph Overview

2.1 Data Format

Per its official description, WordNet's database is made available in:

> *... an ASCII format consisting of eight files, two for each syntactic category. Additional files are used by the WordNet search code but are not strictly part of the database.... Each index file is an alphabetized list of all the words found in WordNet in the corresponding part of speech. On each line, following the word, is a list of byte offsets (synset_offset s) in the corresponding data file, one for each synset containing the word.... Pointers are followed and hierarchies traversed by moving from one synset to another via the synset_offset s.*[2]

The two files for each syntactic category refer to a *data* and an *index* file, with the *data* file holding attributes and relationships of each word in WordNet and *index* containing the mapping of words to synsets present in the *data* file. These relationships and indices are defined as byte offsets, which have the advantage of allowing for APIs working with the WordNet files to quickly traverse the datafile at

[1] http://wordnetweb.princeton.edu/perl/webwn

[2] https://wordnet.princeton.edu/frequently-asked-questions

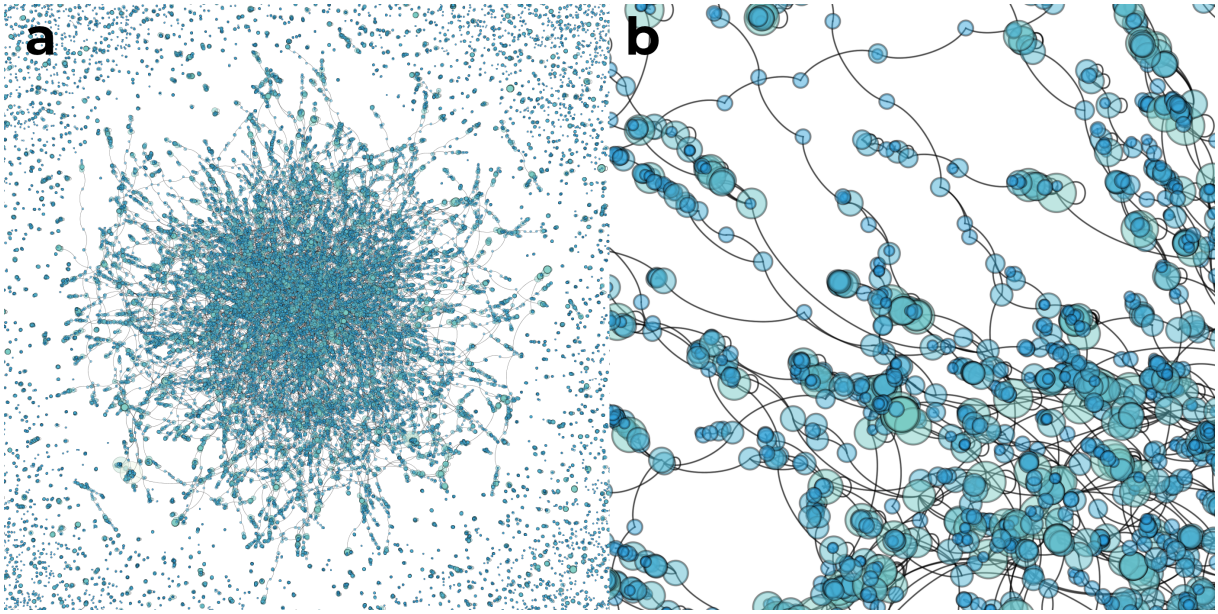

Figure 2: **(a)** The core constellation of all wordsense-wordsense relationships within WordNet, with nodes colored and scaled according to degree (darker and smaller = lower degree; brighter and larger = higher degree). In addition to the central network containing interconnected wordsenses, there exist many disconnected sub-networks that appear when synsets are not rendered. Several of these isolated networks are visible surrounding the main network. **(b)** A detailed view of the northwest corner of the graph in (a), with closely clustered neighborhoods visible as well as long chains of wordsenses that link otherwise disparate regions.

query-time, but come at the cost of being an unintuitive relationship-building and indexing scheme for humans. Further, the number of columns in each row within both *index* and *data* files are variable; for instance, a *data* entry with 3 synset-to-synset relationships will have 8 more columns than an entry with 1 such relationship, as each relationship introduces 4 new fields. This design choice makes the data terse, but increases user effort to parse the structure of and relationships within each row when loading into relational formats, graph databases, and desktop network analysis software.

WAFFLE parses the data within the *data* file and reformats the results into graph representations that trade off representation compactness (previously optimized for quick on-disk or in-memory lookup) for human-legibility and advanced graph analysis when loaded into supported tools (c.f. Section 2.2). The essential form of the transformed data format is that of a node list and edge list, output as .CSV and .JSON files. These files catalog the attributes (e.g. type, part-of-speech, definition, example sentences) and relationships (i.e. source node, type, edge attributes, and target node) of each item in WordNet, respectively. This representation is ready for import in such network analysis tools as Gephi (Bastian et al., 2009) and Cytoscape.[3] For additional utility, WAFFLE also exports the graph as a single .graphml file, a widely-supported graph interchange format that contains both node and edge information (Brandes et al., 2002).[4] WAFFLE is designed to work with the version 3.3 data provided on GitHub under Apache 2.0 license by the maintainers of the Natural Language Tool Kit ("NLTK") (Bird et al., 2009), but is compatible with any data following the WordNet specification.[5] In the spirit of open source software and compatible with the original Wordnet 3.0 license, we present all original components of WAFFLE on GitHub under an open MIT license.[6]

2.2 Construction Methodology

WAFFLE runs in Python 3 (Van Rossum and Drake, 2009), and combines a custom parser for the WordNet format with auxiliary functions to construct an in-memory graph using the NetworkX library (Hagberg et al., 2008). The resulting graph contains

[3] https://cytoscape.org

[4] Specification found at https://graphml.graphdrawing.org/specification.html

[5] https://github.com/nltk/wordnet

[6] https://wordnet.princeton.edu/license-and-commercial-use, and https://opensource.org/licenses/MIT

approximately 288,000 nodes and 392,000 relationships between them, and offers a starting point for the application of any of NetworkX's network-level analysis algorithms, including clustering, centrality, link prediction, graph cutting, similarity, and shortest pathfinding families of operations.[7]

The graph construction begins with an initial parse of the *data* files, loading into memory synset and word attributes from each line sequentially through each part-of-speech's own file. For each line, a dictionary is constructed to hold the synset data with the following structure and key attributes:

- offset: byte offset for lookup of relationships
- type: part of speech type
- words: a list of word dictionaries containing
  - lemma
  - sense: the numerical representation of which use of the lemma the synset describes
- nPointers: number of outbound pointers the synset has
- pointers: a list of pointer dictionaries containing
  - symbol: the WordNet specified symbol representing relationship type (enumerated in Table 1)
  - offset: the target byte offset of the pointer
  - pos: part of speech of the pointer
  - source/target: a special 4-digit hexadecimal designation from WordNet that determines the specificity of the relationship, e.g. from a certain word-sense belonging to this synset to another word-sense, or from the synset to another synset.

This initial traversal and load from the *data* files creates a full representation of WordNet in such a way that synset, word, and lemma relationships can all be individually output and relationships traced to one another without the need for reference to the *index* file. The second pass through involves the building of a node list containing all synset and word information as well as an edge-list containing relationships by name rather than by offset. If, however, the *source/target* field of a pointer designates that the relationship is from a synset to a wordsense, wordsense to synset, or wordsense to wordsense, it is possible that the reference is to a wordsense that is known only by its offset and not yet its identity (the synset it belongs to may be later in the file). Consequently, the synset-to-wordsense memberships are stored in a separate dictionary and relationships that belong to this category are saved. Once the synset-to-synset relationships are all constructed and the file iteration complete, the remaining relationships are traversed and edges created. This results in a complete, byte-offset-resolved data format. As synsets have no representation other than their conceptual meaning, they are identified by their offsets as primary keys in the WAFFLE graph.

At this stage, the output is in its most flexible form and users looking for maximum versatility should take the .CSV and .JSON node list and edge list outputs as starting points for their work. For users interested in analyzing WordNet within Python, WAFFLE also constructs a NetworkX graph from the in-memory representations of this data, annotating edge labels and weights that are exported into a .graphml format for graph transformations and further manipulation.

2.3 Graph Overview and Summary Statistics

The WAFFLE graph contains 117,478 nodes of type *synset*, 170,479 *wordsense*, and 391,949 edges spanning membership relations and 26 other semantic relationship symbols. Table 1 provides a breakdown of total edges in the network by relationship type, and Figure 2 illustrates a top-level look at the information content of WordNet's relationships. Figure 3 showcases the different presentations of the WordNet data that subgraph extracts and transformations on the base graph structure can provide.

2.4 Graph Transformations

The graph of synset-to-synset, wordsense-to-wordsense, and synset-to-wordsense relationships across the 27 relationships (Table 1) represents the most heterogeneous form of the WordNet graph. While this form is a good starting point for familiarizing with WordNet's structure, it can be useful to condense either multiple edge types together or represent parallel edges as one, import into a database-specific format, or study only wordsense-to-wordsense relationships wherein a common shared synset induces an edge between wordsenses. WAFFLE provides avenues for these transformations, each of which serves as a template for further user-driven customization.

[7] https://networkx.github.io/documentation/stable/reference/algorithms/index.html

Symbol	Relationship	Count
has_member	Synset Membership	208353
~	Hyponym	89174
@	Hypernym	89174
+	Derivationally Related	74591
&	Similar To	21434
#m	Member Holonym	12288
%m	Member Meronym	12288
%p	Part Meronym	9111
#p	Part Holonym	9111
@i	Instance Hypernym	8587
~i	Instance Hyponym	8587
\	Pertainym to Noun or Derived from Adjective	8054
!	Antonym	7983
-c	Domain Member Topic	6689
;c	Synset Domain Topic	6689
^	Also See	3276
$	Verb Group	1744
-r	Domain Member Region	1498
;r	Synset Domain Region	1498
-u	Domain Member Usage	1368
;u	Synset Domain Usage	1368
=	Attribute	1278
#s	Substance Holonym	797
%s	Substance Meronym	797
*	Entailment	408
>	Cause	221
<	Participle of Verb	73

Table 1: WordNet relations and counts in the WAFFLE graph. Availability of relations is keyed to part of speech (https://wordnet.princeton.edu/documentation/wninput5wn).

2.4.1 Edge Condensation and Weighting

WAFFLE provides an optional step in the graph creation process that normalizes each of the 27 semantic relationships (many are directional inverses of one another – e.g. hypernymy and hyponymy) into a single edge type of connectedness, and stores the count of relationships condensed between any two nodes in the graph as the weight between them. Although this edge condensation certainly results in a reduction of total information content, it presents the advantages of edge normalization and creation of bidirectionally-weighted edges between nodes. This makes the treatment of the graph as homogeneous in centrality and betweenness calculations more immediately accessible.

2.4.2 Graph Database Import

Although NetworkX and Python provide a powerful platform for in-memory graph creation and analysis, potential users of WordNet may be interested in loading and querying WordNet from within a user's graph database. To this end, WAFFLE produces a .graphml output that is ready for import into a graph database, and includes a Cypher query-language script for importing WordNet into Neo4j, a prominent desktop and server-deployable graph database, using its officially-supported APOC (Awesome Procedures On Cypher) plugin.[8] This enables the WAFFLE-produced graph to be readily-queryable by local and remote applications as well as data analysts issuing Cypher. Analyzing WordNet through the use of a powerful graph query language like Cypher opens the door to direct path-based querying of the data, as illustrated in the Figure 3.

2.4.3 Expanded Graph Flexibility

For the analysis of words in specific senses and their relations to one another, users may only want to consider synsets as stepping stones to and from specific wordsenses, and in so doing analyze their relationships only by proxy. To achieve this, a transformation of the graph through the following Cypher can be conducted:

```
MATCH (w1:Wordsense)-[:has_member]-(s:
    Synset)-[:has_member]->(w2:Wordsense)

WHERE id(w1) > id(w2)
MERGE (w1)-[:shared_synset]-(w2)
```

From this point, the direct wordsense-to-wordsense relationships can be explored and subgraphs extracted, providing an intuitive perspective towards exploring semantic relations of words and their shared meanings. This approach condenses the total number of synset-membership-based edges in half (each new edge represents two original connections), optimizing the memory footprint and query structure.

Compared to the full synset-inclusive graph, this representation is both visually-accessible and enables wordsense-to-wordsense pathfinding that neither the original graph nor references to the WordNet *data* and *index* files provide (directly or in a specific API). A count of degrees of separation in this graph of directly-linked words translates simply to how many synsets (or other direct connections) away from one another the two words are. Similarly, by abstracting away the synset-to-synset relationships, users of this particular view do not need to resolve polarity, or semantic directionality, of the many synset-to-synset relationships and can focus on the introduced necessarily-equivalent "shared_synset" relationships. Although

[8] The pure Cypher (non-APOC) components of this workflow are applicable as well to any database supporting the Cypher language. See https://www.opencypher.org and https://www.neo4j.com

direct wordsense-to-wordsense relationships encoded in WordNet are retained by this example transformation, users can modify the Cypher or remove these connections before their analyses to ensure homogeneity of edge-types. As a result, node-level (e.g. degree, betweenness centrality) statistics and neighborhood (e.g. community detection) operations can be produced where each edge is directly comparable to all other edges in the network. Both this format of the WordNet graph and the synset-inclusive form are available with the WAFFLE source code (c.f. Section 5).

2.4.4 Subgraph Extraction

Subgraph extraction using WAFFLE enables focused views such as the examples in Figure 3. This process is useful not just for creating publication-ready graphics, but also for targeted exploration of specific regions of the full WordNet. A traditional example of subgraph extraction involves selecting a seed group of nodes and including nodes isometrically from that core. More creative and specialized subgraphs such as those containing all nodes and induced edges within one degree of the shortest spanning path between two wordsenses or synsets can be created as well. This flexibility in navigating and observing the WordNet graph through WAFFLE not as a tree structure but as a non-rooted graph structure offers unique opportunities. These comparisons, as well as a treatment of analogous functionality, are featured in Section 3.

3 Features, Strengths, and Comparisons

While novel in structure, WAFFLE parallels but does not present itself as a replacement to existing representations and forms of access to WordNet. For comparison, we present several canonical operations performed on WordNet through its WAFFLE-processed form and as accessed through NLTK. We break common functionality associated with WordNet into: (1) Information Retrieval; (2) Synset Relationship-finding; (3) Computation of Semantic "Distance"; and (4) Visualization. For each category, we present examples and syntax in both WAFFLE and the NLTK WordNet *wn* library.[9]

3.1 Information Retrieval

Lookup operations treat WordNet as an information repository rather than a structure or tool for computation, and accordingly, stand to suit common methods of information retrieval just as well as graph-based approaches. Despite their simplicity, these lookups are a very natural place to begin investigation of linguistics using WordNet and offer a direct comparison between NLTK-equivalent standalone *wn* WordNet API and WAFFLE. For ease of reproduction and generalization, several WAFFLE graph examples are provided as Cypher queries.

Synset lookup by lemma in NLTK returns a list of Sysnset objects that correspond to the called lemma, in this case `wn.synsets('establishment')`, notably including synsets that do not have a wordsense corresponding to the lemma queried:

```
Synset('constitution.n.02')
Synset('institution.n.01')
Synset('administration.n.02')
Synset('establishment.n.04')
...
Synset('establishment.n.07')
```

This can be attributed to the way that *wn* identifies these synset lookups, ordering them based on frequency counts from WordNet concordance texts.[10]

By comparison, the equivalent Cypher query:

```
MATCH (w:Wordsense)-[:has_member]-(s:
    Synset) WHERE w.lemma = "
    establishment" RETURN s
```

is more verbose, but precisely describes the relationship between what's being matched (a wordsense with the exact lemma) and what's being returned (a synset with membership relation). Because of the design of the WAFFLE graph, synsets are identified not by a single exemplar usage, but by a unique identifier corresponding to the synset's original byte offset. This trade-off reduces the opportunity for synset misinterpretation, and all wordsenses belonging to a synset can be retrieved through an inversion of the original query:

```
MATCH (s:Synset {id:someID})-[:
    has_member]-(w:Wordsense) RETURN w.
    lemma
```

This operation is done in *wn* by calling the `.lemmas()` function of a Synset object.

This theme of terseness being exchanged for flexibility and explicitness in WAFFLE continues for definition and example lookups on synset objects. These operations are handled by the

[9] https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html

[10] https://wordnet.princeton.edu/documentation/wn1wn

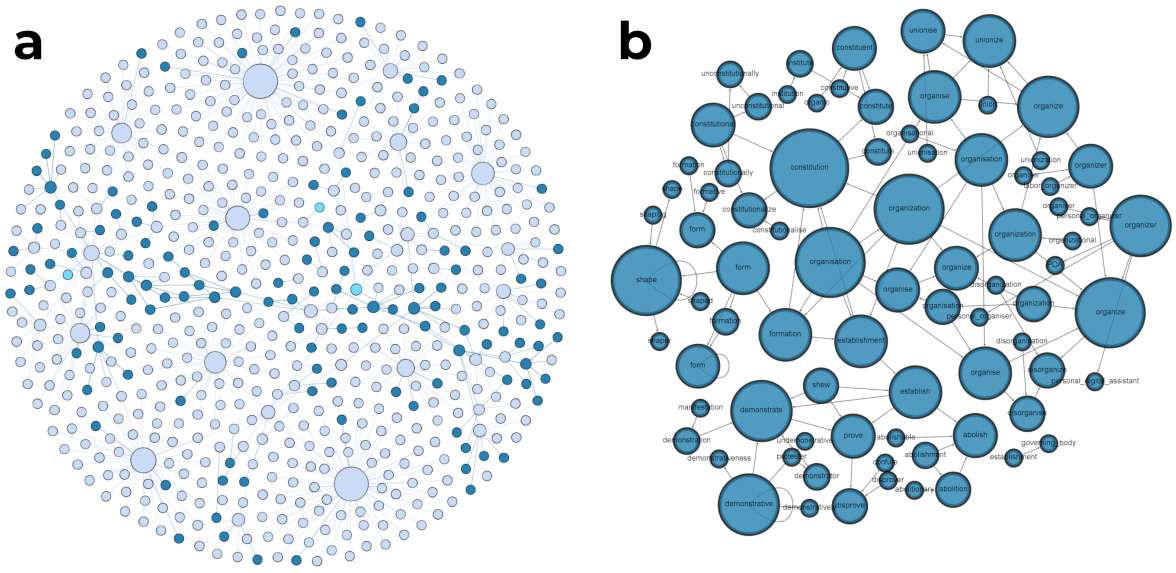

Figure 3: For the lemma *establishment*: **(a)** The uncondensed graph view of any wordsense (turquoise) with the lemma and all connections within 3 degrees of those wordsense nodes. The direct relationships of wordsenses (dark teal) to one another is not very clear, but the overall topology and role of synsets (light blue-grey) in bridging relationships is made clear. **(b)** The condensed wordsense-only graph view of wordsenses and all *shared_synset* and syntactic relationships within 4 degrees of those nodes. The connections between individual wordsenses is much more clear compared to (a), but senses that do not have any direct relationships to another wordsense or shared synsets – in this case one of the three wordsenses – are not included in this transformation. Node size is modulated by degree.

`.definition()` and `.examples()` function calls in *wn*. For WAFFLE, these values are stored as properties (attributes) of the synset object nodes in the graph and closely align to the style of retrieval in *wn*. For example, the Cypher query used to retrieve the definition of any word with its associated numerical synset identifier (n874164 - *demonstrative_0*) is:

```
MATCH (s:Synset {id:someID}) RETURN s.
    definition
```

Considering these are attribute-lookup operations rather than path-traversal operations, no graph pattern matching is required. However, in the event that *all* definitions for synsets containing a certain lemma in their word sense were to be investigated, the associated Cypher query combines two *wn* calls and organizes results into a single query. Returning to our example, in order to look up all definitions for synsets that contain words of the lemma "establishment", the Cypher would be:

```
MATCH (w:Wordsense)-[:member\of]-(s:
    Synset) WHERE w.lemma ="
    establishment" RETURN s.definition
```

As the desired lookup becomes more complex, the value of concisely stating the information retrieval task as a graph look-up begins to quickly outweigh the original advantage in terseness a traditional interface to the data offers.

3.2 Synset Relationship-finding

Relationship-finding is an operation that graphs are configured to perform, and graph-querying languages like Cypher designed to express. As a result, WAFFLE presents a method of working with familiar relationships and introduces the ability to easily specify graph traversals in WordNet that would otherwise have recalled many individually-chained or recursive function calls.

In practice, single-level depth relationship operations in *wn* are straightforward: individual functions belonging to the Synset object will return the result of the immediate neighbor lookup - e.g. Table 2. In the WAFFLE graph, the following query illustrates beginning at a target wordsense and finding synsets related to its parent synset – in this case with the @ (hypernym) relationship.

```
MATCH (w:Wordsense)-[:has_member]-(s1:
    Synset)-[:@]-(s2:Synset) RETURN s2
```

```
MATCH p=(w:Wordsense)-[:has_member]-(s1:
    Synset)-[:@]-(s2:Synset) RETURN p
```

NLTK	*return*
.hypernyms()	[body.n.02]
.hyponyms()	[county_council.n.01, curia.n.01, executive.n.02, government_officials.n.01, management.n.02, judiciary.n.01, top_brass.n.01]
.part_holonyms()	[government.n.01]
.member_meronyms()	[advisory_board.n.01]

Table 2: Example NLTK *wn* relation calls and returns for the Synset object "establishment.n.03".

Any of the relation symbols (c.f. Table 1) can be substituted in the query to replace the hypernym relationship with the semantic relation of interest. It is worth noting that two of the relationships share the \ relationship symbol, which represents either a noun pertainym or derivation from an adjective based on context and results in a total of 26 unique syntactic edge labels within the WAFFLE graph.

3.3 Computation of Semantic "Distance"

NLTK's *wn* provides a number of distance operations to quantify differences between shared synsets (common in wordsense disambiguation and document/query similarity tasks). In general, methods for computing path similarity range from simple (number of hops in a hierarchy) to complex - e.g. Leacock-Chodorow (Leacock and Chodorow, 1998) or Wu-Palmer (Wu and Palmer, 1994) similarities in Table 3.

NLTK	*return*
Lowest Common Hypernym [Subsumer] .lowest_common_hypernyms()	*entity.n.01*
Shortest Path (1/number of hops) .path_similarity()	*0.0833*
Leacock-Chodorow similarity .lch_similarity()	*1.1526*
Wu-Palmer similarity .wup_similarity()	*0.1538*

Table 3: NLTK path similarities for comparing the Synset objects wn.synset("establishment.n.03") and wn.synset("establishment.n.04").

These functionalities are unique to *wn* because of its model of WordNet as a tree structure, where traversals up and down the tree – up to and including the root nodes that bind each conceptual category (e.g. "entity") – provide markers of similarity and distance. In WAFFLE, there is no concept of moving up or down individual hierarchies; instead, these traversals in granularity and specificity represent directional edges in the graph. Distance queries, configurable to report on only one type of edge or multiple edge types, can be used to find path lengths from synsets or wordsense to one another, but these results would be incomparable to the specific calculations underlying each similarity or lowest-common ancestor lookup.

The same superimposition of multiple WordNet hierarchies that makes WAFFLE directly incomparable to existing similarity measures offers a novel approach to similarity and pathfinding in the WordNet data: the identification and exploration of cyclic structures in WordNet is now explicitly defined. Furthermore, WAFFLE's flexibility to be transformed on a graph level is unmatched by *wn*. The transformation in Section 2 is but one example of modifying the base graph structure to create a purpose-built representation.

3.4 Visualization

Building and visualizing graph structures using NetworkX and Matplotlib in *wn* is *possible*.[11] However, the nature and scope of these created graphs is tied to and limited by the funnel of the API design. For *wn*, an example of this limitation is that each graph query must involve recursive calls to the relationships branching out from a word, lemma, or synset of interest. In contrast, fluid and customizable graph visualization is one of the foremost design principles behind the structure and format of WAFFLE.

Graph data structures lend themselves naturally to network visualization, and the provision of multiple data formats in common interchange formats and specific scripts for loading and transformation in Cypher-enabled platforms creates a platform on which all users are invited to explore and expand. Figures 2 and 3 are but introductory examples of the types of visualization that WAFFLE can be used to generate in the study of linguistic relations and the structure of language in general.

4 Related Work

There are many ways of accessing the content WordNet beyond the the database files which allows for a tremendous amount of choice in developing against WordNet.[12] Further, NLTK and comparable packages from WordNet::Similarity (Pedersen et al., 2004) and spaCy, to name a few, provide

[11] Bird et al. (2009, 170-171) and https://www.nltk.org/book/ch14.html

[12] https://wordnet.princeton.edu/related-projects

comprehensive approaches to exploring not only the content, but also the structure of WordNet.[13] However, compared to WAFFLE, these existing offerings facilitate one-off to moderately-scalable investigations; given more to exploratory research and processing rather than use with large data sets, sophisticated applications, or classes of problems that need to leverage the structural elements of WordNet.

Some existing offerings *are* more oriented to graph-based structures. For example, the Global WordNet Association does produce formats (JSON, XML, RDF) that could be easily translated into graph data structures but introduce a number of additional relationships that reflect ongoing research and linkage to multi-lingual WordNets.[14] Similarly, FrameNet (Baker et al., 1998) is graph-based and accessible with NLTK, but provides more semantic and syntactic connections within the context of frame semantics with no direct link to WordNet. ConceptNet (Speer et al., 2017) is graph-based as well with a closer relationship to WordNet with explicit external linking, but is focused on a much broader range of information for natural language understanding, common sense reasoning, crowd sourced knowledge, etc. Future connections to these offerings will be explored, but, as is introduce a number of additional complexities that WAFFLE seeks to avoid.

5 Availability and Future Work

We have presented WAFFLE, an open source graph data structure that relies upon platform-agnostic formats to facilitate robust interrogation and flexibility when using WordNet in research or applications. WAFFLE's software, example load scripts, and the associated figures and graph files are available at `github.com/TRSS-NLP/WAFFLE`.

While we encourage users to capitalize on the advantages afforded by the design and transformations presented in Section 2 to implement the WordNet data in entirely new ways, we envisage several avenues of augmentation: (1) linking WAFFLE to corpora to perform more sophisticated path measures using information content (Jiang and Conrath, 1997; Lin, 1998; Resnik, 1995) and associated word embeddings; (2) connecting additional WordNet information such as morphosemantic links, lexical forms and other semantic annotations (existing as "standoff" files; and (3) multi-lingual connections through with Open Multilingual Wordnet and potentially others referenced in Section 4.[15]

[13] `https://spacy.io/universe/project/spacy-wordnet`

[14] `http://globalwordnet.github.io/schemas/`

Acknowledgments

Thank you to Peter Chang, Ian Coffman, Saul Dorfman, Andrew Follmann, Eleanor Hagerman, and Spencer Torene for early feedback and multiple reviews. Thank you also to three anonymous reviewers from NLP-OSS for suggested improvements and constructive comments which improved the final version of this paper.

References

John Atkinson, Anita Ferreira, and Elvis Aravena. 2009. Discovering implicit intention-level knowledge from natural-language texts. *Knowledge-Based Systems*, 22(7):502–508.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING-ACL 98*, pages 86–90.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Jean-Philippe Bernardy and Aleksandre Maskharashvili. 2019. Two experiments for embedding wordnet hierarchy into vector spaces. In *Proceedings of the Tenth Global Wordnet Conference*, pages 78–84. Oficyna Wydawnicza Politechniki Wrocławskiej.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, Martín López-Nores, Jorge García-Duque, Ana Fernández-Vilas, Rebeca P. Díaz-Redondo, and Jesús Bermejo-Muñoz. 2008. A flexible semantic inference methodology to reason about user preferences in knowledge-based recommender systems. *Knowledge-Based Systems*, 21(4):305–320.

Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and Marshall S. Marshall. 2002. Graphml progress report: Structural layer proposal. In *Proceedings of the 9th International Symposium Graph Drawing (GD2001, LNCS 2256*, pages 501–512. Springer-Verlag.

[15] See `https://wordnet.princeton.edu/download` and `https://compling.hss.ntu.edu.sg/omw/`

Henrik Bulskov, Rasmus Knappe, and Troels Andreasen. 2002. On measuring similarity for conceptual querying. In *Proceedings of the 5th International Conference on Flexible Query Answering Systems*, FQAS '02, page 100–111, Berlin, Heidelberg. Springer-Verlag.

Yufeng Diao, Hongfei Lin, Di Wu, Liang Yang, Kan Xu, Zhihao Yang, Jian Wang, Shaowu Zhang, Bo Xu, and Dongyu Zhang. 2018. WECA: A WordNet-encoded collocation-attention network for homographic pun recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2507–2516, Brussels, Belgium. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th Research on Computational Linguistics International Conference*, pages 19–33, Taipei, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACL-CLP).

Łukasz Kobylinski and Michał Wasiluk. 2019. Deep learning in event detection in polish. In *Proceedings of the Tenth Global Wordnet Conference*, pages 216–221. Oficyna Wydawnicza Politechniki Wrocławskiej.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283.

Yuhua Li, Zuhair A. Bandar, and David Mclean. 2003. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann.

Daniel Loureiro and Alípio Jorge. 2019. Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41.

Tomasz Naskręt, Agnieszka Dziob, Maciej Piasecki, Chakaveh Saedi, and António Branco. 2018. Wordnetloom – a multilingual wordnet editing system focused on graph-based presentation. In *Proceedings of the Ninth Global Wordnet Conference*, pages 191–200. Singapore.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2).

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'03, page 241–257, Berlin, Heidelberg. Springer-Verlag.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145.

Yuval Pinter and Jacob Eisenstein. 2018. Predicting semantic relations using global graph properties. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1751, Brussels, Belgium. Association for Computational Linguistics.

Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. 1989. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, page 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the 31st Association for the Advancement of Artificial Intelligence Conference (AAAI 31)*, pages 4444–4451. AAAI.

Rohini K. Srihari, Zhongfei Zhang, and Aibing Rao. 2000. Intelligent indexing and semantic retrieval of multimodal documents. *Inf. Retr.*, 2(2–3):245–275.

Mark Stevenson and Mark A. Greenwood. 2005. A semantic approach to ie pattern induction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 379–386, USA. Association for Computational Linguistics.

Ali Ghobadi Tapeh and Maseud Rahgozar. 2008. A knowledge-based question answering system for b2c ecommerce. In *Proceedings of Fifth International Conference on Information Technology: New Generations (ITNG 2008)*, pages 321–326.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2019. Sense vocabulary compression through the semantic knowledge of wordnet for neural word sense disambiguation. In *Proceedings of the Tenth Global Wordnet Conference*, pages 108–117. Oficyna Wydawnicza Politechniki Wrocławskiej.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, page 133–138, USA. Association for Computational Linguistics.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1374–1385, Osaka, Japan. The COLING 2016 Organizing Committee.