Extracting Symbolic Knowledge from WordNet Glosses

Sonya Anopa

School of Computer Science

Carnegie Mellon University

I. Introduction

The Scone project seeks to provide an open-source knowledge-base system for common-sense reasoning and language understanding [5]. It stores symbolic knowledge which could be either general or application specific and allows for inclusion within other software applications that might need a knowledge base. The currently available version includes some core knowledge; however, adding new knowledge is mostly done by hand, which is a tedious and time-consuming process.

WordNet is a lexical database for English that organizes words into sets of synonyms ("synsets") and establishes a number of relations between them within its structure [7]. The most important is the hypernym-hyponym relation, relating two concepts as being more general or more specific in relation to each other, stating that the more specific concept inherits from the properties of the more general one. WordNet also provides glosses with the synsets, which are human-readable definitions for the synsets. They encode additional knowledge about these concepts that is not represented within the WordNet structure.

Therefore, WordNet glosses provide a great source of information that can be extracted and imported into Scone to expand its knowledge automatically. This research project "reads" WordNet glosses to extract useful symbolic knowledge structures that can be added to Scone, which is achieved by using dependency parsing and identifying structures that can be extracted, as well as their equivalent structures in Scone.

Similar work has been done by James Allen et al., extracting information from WordNet glosses, but targeting a description-logic system [1]. Their approach consisted of parsing the definitions such that they generated a logical form graph encod-

ing relations between concepts and defining concepts within their definitions. Next, the graph was translated into OWL, the Web Ontology Language, specifically its description-logic version, by creating concepts from the graph and defining classes in OWL. As Scone is not a first-order-logic system, which description logic builds on, this approach cannot be used directly or with little modification.

This project creates a Python script that implements the approach described below for extracting the knowledge from WordNet and a generalized script that allows extracting the same type of knowledge from any definitions.

II. Approach

A. Method

The approach taken to the problem can be described in several steps, each of which will be explained later in detail.

1. Extract all hypernym-hyponym relations encoded and create is-a links between these concepts in Scone.

2. Extract all meronymy-holonymy relations and use part of, member of, and makes up relations to represent part, member, and subtance relations.

3. Extract head of dependency tree with any constricting dependencies and add another is-a link.

4. Extract any constructs that describe the concept using different terminology and connect using eq links.

5. Extract constructs that can be converted into Scone relations and Scone roles and import them.

B. Extracting Hypernym-Hyponym Relations

WordNet can be visualized as a directed graph of links that connect more general concepts - hypernyms - to more specific concepts that inherit from the general concepts - hyponyms. While it is possible to read WordNet data directly from the data files WordNet provides, NLTK, a Python toolkit, allows easier access to such data by functioning as an API that queries WordNet and returns the necessary information [2]. We use NLTK in this capacity by traversing through the WordNet ”graph” and finding all the hypernym-hyponym pairs. The corresponding structure in Scone is an is-a link that connects two Scone concepts. Thus, every pair of concepts is added to Scone as new types if they do not exist yet and then connected with an is-a link from hyponym to hypernym. The statement ≪A is-a B≫ means that A inherits all the properties of B. An example transformation is shown in Figure 1 (note: curly braces indicate the Scone syntax for concepts).

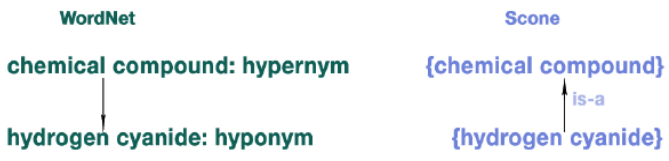

Figure 1. Extracting hypernym-hyponym relations

C. Extracting Holonymy-Meronymy Relations

WordNet also includes holonymy-meronymy relations as part of its structure [7]. Holonymy is a relation between two concepts X and Y such that Y's are parts of X; meronymy is the inverse relation. We also use NLTK to obtain these relations and create as corresponding structures Scone relations. Relations in Scone allow representing richer information about the relationship of one concept to another than just by using is-a links; relationships can be user defined [5]. WordNet differentiates between part, member, and substance holonymy/meronymy: these three turn into {part of}, {member of}, and {makes up} relations in Scone, respectively. After defining the relations, pairs of elements from WordNet are connected by a statement in Scone stating that the relation exists between these elements, e.g. {engine} {part of} {car}. In this case, the concepts participating in the relationship have already been added to Scone and thus it is not necessary to check whether they exist already or not. Only meronymy relations are added to Scone because Scone allows indicating the existence of an inverse relation within one statement.

D. Syntactic Dependencies

Prior to moving on to the next section, it is important to establish a background in dependency parsing, as that is the main component of the approach in steps 3-5 of the method.

In dependency grammars, a sentence's syntactic structure can be described as consisting of words and binary relations between those words [6]. This description can be represented using a directed graph with labeled edges with nodes as words and edges as dependencies. An example of a dependency parse can be seen in Figure 2. In a typed dependency parse, the edges are labeled from a predetermined set of dependencies. We use the Stanford CoreNLP toolkit, which provides a rule-based dependency parser developed by de Marneffe, MccCartney, and Manning, and a neural-network dependency parser developed by Chen and Manning [4], [3]. Both of these parsers use Universal Dependencies to tag the grammatical relations, which is a specific set of dependencies designed to facilitate multilingual dependency parsing [10].

Dictionary definitions are more constrained than regular sentences. In particular, they tend to contain the main word that describes a concept as the head of the sentence when parsed into dependency relations; for example, in WordNet "hydrogen cyanide" is defined as a "highly poisonous gas that ...".

When parsing this definition, the word ”gas” will be the head of the resulting graph, which is the main concept that ”hydrogen cyanide” is defined as. Therefore, adopting this assumption we can rely on this property to extract the main concepts that define WordNet synsets and other information. Of course, the assumption is not always true; to eliminate such cases some additional processing is performed that is described in the next section. Other dependencies of the head word provide a way to restrict the range of information or to introduce additional information about the concept being defined. We use both of these types of information.

Overall, the approach in steps 3-5 of the method relies on finding the dependency graphs of WordNet definitions, extracting the useful structures defined later from these graphs, and importing corresponding structures to Scone.

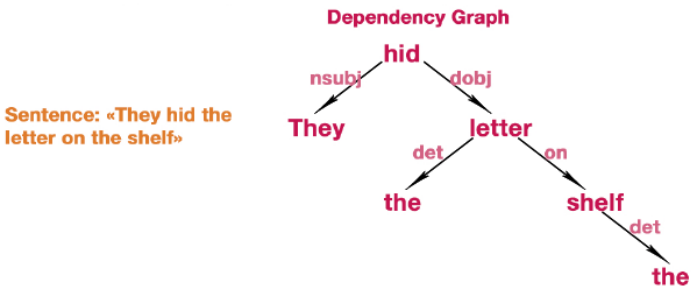

Figure 2. A dependency graph example from [6].

E. Extracting Parents of Concepts

Scone allows several is-a links to be attached to a concept, so the parent concept extracted from the WordNet structure can be complemented with a parent concept extracted from the definition of the concept [5]. As the definition is a human-readable and human-understandable one, the concept that a concept is described in terms of more likely encodes common-sense knowledge better than the highly-detailed WordNet structure; therefore, it makes sense to add that concept with an additional is-a link as well.

As noted previously, the parent is taken to be the head of the dependency graph that results from parsing the definition. As dependency parsers are not perfect, two dependency parsers are used to identify the graph. If they agree on the head of the graph, either result can be used. Otherwise, if they disagree, we find that usually at least one of the parsers produces the correct answer. Because the WordNet synsets we are looking at are nouns, we expect the parent concept to be also a noun (i.e. concepts are defined in terms of other concepts, not adjectives, verbs, or other parts of speech). Therefore, if the parsers disagree and one of them has a noun at the head of the graph, we choose that parse. In other cases we skip this synset to be safe, as we want to limit the number of potential errors introduced into Scone. Then we add both the WordNet synset we are examining and the parent concept into Scone if they are not yet present and add another is-a link between them. A small improvement of this process uses

the 'compound' relation to extract all the parts of a compound word such as "phone book" and use that compound word as the parent concept instead [10].

To make the parent concept as specific as possible, we also extract adjectives "around" the head of the parse tree and create Scone intersection types. Intersection types are concepts created from a set-theory-style intersection of other concepts. In other words, for any concepts that create an intersection-type concept, it is defined to be all of them at once. For example, returning to the synset "hydrogen cyanide," the relevant part of its definition is "poisonous gas". The previous section describes the way to extract "gas" and state that "hydrogen cyanide is-a gas" in Scone. However, "poisonous gas" is a more specific concept; to that end, we extract "poisonous" and "gas," create corresponding concepts in Scone, create an intersection type of "poisonous gas," and finally create the link "hydrogen cyanide is-a poisonous gas." Figure 3 depicts this transformation.

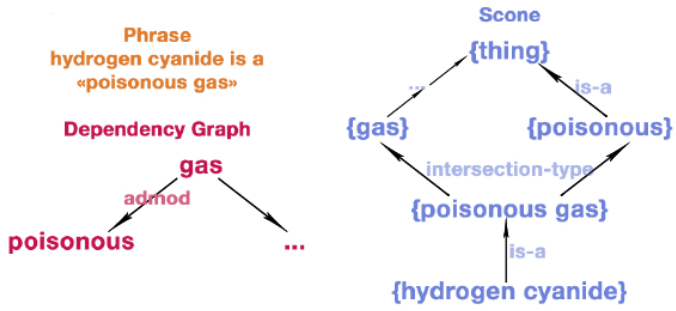

Figure 3. Extracting parents of concepts and creating intersection-types.

1) Word Sense Disambiguation: In the previous example, {gas} has {thing} listed as the eventual parent of the concept, since {thing} is the root node of the Scone hierarchy, but what is the immediate parent? When we extract "gas" from the definition, we need to connect it to as specific of a parent concept as possible. Otherwise, we are improving the problem only by one level: connecting "gas" to "thing" directly tells us later on that "hydrogen cyanide" is a "gas" rather than something unknown, but "gas" is essentially unknown.

Finding a specific parent relies on WordNet as well. Namely, once we have traversed the entire WordNet graph and extracted the hypernym-hyponym relations, "gas" can possibly be identified as a specific WordNet synset. Alternatively, if the concept already exists in Scone, that concept can be used. If neither is an option, our backup action is to add it into Scone with "thing" as the parent.

Identifying a word as a specific WordNet synset means searching for synsets that have that English common name in WordNet, which can be easily done with NLTK, and then identifying which sense of the word is used. Word senses are the different meanings a word can be used in. For example, "gas" has 6 WordNet senses: a gaseous state, a fluid in the gaseous state, gasoline, flatulence, the gas pedal in a car, and natural gas.

This problem is generally called word sense disambiguation. While it is not solved in general, some can be completed when it is constrained enough, as it is in this case. The process we follow to identify the sense is below:

1. If there is one sense of the word in WordNet, pick that sense.

2. As the concept being searched for defines a WordNet synset, it should be some superclass of the synset. If one of the senses appears on the synset's path to the WordNet root, pick that sense.

F. Extracting Appositional Modifiers

Another dependency that has a corresponding representation in Scone is the appositional modifier dependency, or 'appos' in Universal Dependencies specification. According to Universal Dependencies, an appositional modifier of a noun is "a nominal immediately following the first noun that serves to define or modify that noun" [10]. In other words, a phrase labeled as being an appositional modifier describes the parent word using some other word phrase; the two are equally informative descriptions of the overall concept.

In Scone this case is represented by using an eq link [5]. The method for importing this structure is thus as follows: after identifying the head of the dependency graph in the definition, we can extract any 'appos' dependencies attached to that head word, import them into Scone as a whole concept, and add an eq link between that concept and the concept corresponding to the WordNet synset whose definition we were reading. This process is pictured in Figure 4. We note again that we also use 'compound' dependencies to extract the full compound words connected with the 'appos' dependencies.

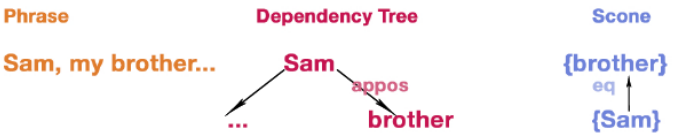

Figure 4. Extracting Appositive Dependencies

Additionally, the extractions described in the previous part that allow creating intersection types can also be included when extracting the appositional modifiers. Thus, rather than simply equating "Sam" and "brother" from the figure above, an intersection type of "my" and "brother" can be created first to then proceed to an eq link between "Sam" and "my brother". Additionally, inserting "brother" into Scone runs into the same problem as described in the previous section - the parent of this concept has to be identified. The word sense disambiguation approach applied previously is also used in this case to solve that problem.

G. Extracting Roles

A role in Scone allows describing some concepts as performing some role for other concepts. For

example, if something is said to be ”the office of the Chair,” it plays the role of the office for the Chair. This construct allows for more powerful inferences that the ones possible with is-a and eq links only. Using it requires creating a new-type-role or a new-indv-role (individual, referring to instances rather than classes - usually a proper noun), ”office” in the example above, and connecting the concept playing the role with the concept it ”belongs” to (”the Chair” above) with a link that is created with the ”x-is-a-y-of-z” function.

In WordNet glosses this idea can be found when the synset whose gloss we are reading is defined as playing a role for some other concept. For example, a brush can be defined as a ”tail of a fox”: so, the brush’s role is being a tail to a fox [9]. In dependency graphs this pattern is represented with nominal modifier and case-marking dependencies, or ’nmod’ and ’case’ in Universal Dependencies notation [10]. Here we can use the ’compound’ dependency to extract compound words as well.

Nominal modifiers, when attached to nouns, represent an attribute of the parent noun. The ’case’ dependency is used to mark words such as prepositions that case-mark the noun they depend on. From above, ”tail of a fox” has ”tail” as the head word in the dependency parse, ”fox” is connected to tail with the ’nmod’ relation, and ”of” is assigned the ’case’ relation. Figure 5 represents this type of extraction. We only extract such patterns if the word connected with the ’case’ relation is ’of’ since that is the way roles in Scone can be represented as English phrases, but other cases can potentially function as roles as well. Additional work is needed for a more detailed approach.

Figure 5. Extracting Roles

It is worth noting that this pattern is less reliable than the ones previously described. While it is quite specific, information that does not represent a role relationship can still have this grammatical pattern. For example, the definition for ”tuft” in WordNet is ”a bunch of feathers” [9]. According to the pattern, we can state that a tuft plays the role of a bunch for feathers, which is incorrect.

H. Extracting Relations

Aside from meronymy/holonymy, WordNet definitions provide another source of relations for Scone. As mentioned before, relations in Scone describe a user-defined relationship between two concepts, e.g. ”taller than,” which can be instantiated with a statement such as ”A taller than B” [5].

A source of such relations in WordNet glosses are adjectival clauses, named 'acl' dependencies within Universal Dependencies; they relate a noun being modified and a clause modifying that noun [10]. A common wording in WordNet glosses is "X VBG Y," where VBG is a verb in present continuous form; for example, podetium is defined as an "organ ... resembling a stalk" [9]. The common sense knowledge we are trying to extract from this definition is the fact that podetium resembles a stalk, along with the other extractions that can be completed following the previous sections. To do so, we extract the 'acl' dependency - in this case, "resembling" and its dependencies, such as 'nmod,' 'dobj,' and 'xcomp,' which refer to words that describe that verb. After some processing with the Pattern Python library that allows us to conjugate the verb into present-tense third-person singular, we create the relation {resembles}. Finally, we create the necessary concepts for the WordNet synset and the word connected with 'nmod,' 'dobj,' or 'xcomp,' using word sense disambiguation again to help with that task, and instantiate the relation by creating a new statement of the form {podetium} {resembles} {stalk}. This process is depicted in Figure 6. This case also uses the word sense disambiguation process described above when adding new concepts into Scone; however, relations do not go through it as, while they can form a relation hierarchy in Scone,

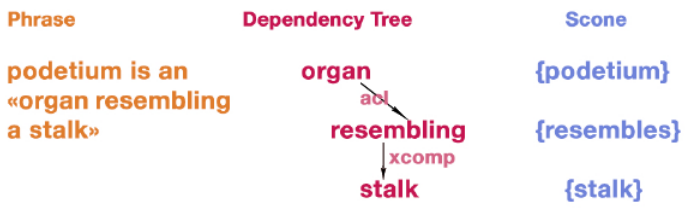

Figure 6. Extracting Relations

that would require a way to identify relationships between verbs, which is out of the scope of this project.

I. *Outcome*

Overall, after going through the steps of the method the script outputs Lisp code that can be loaded into Scone as knowledge. The code contains not only the concept definitions and the created links, but it also contains several functions to make sure Scone knowledge that is already present is not repeated. For example, if the Lisp code needs to create a concept, rather than simply using a Scone function for doing so it calls a separate function that checks whether such an element already exists; if so, any additional properties added from the generated Lisp code are added to that element; otherwise, a new concept is created. A similar procedure is performed for creating new relations, is-a links, and eq links.

III. Generalization

To generalize the tool, we rely on the same method as described above, but excluding WordNet. Given a definition and a word, the generalized script returns the same type of extractions that the previous sections perform, but it does rely on WordNet for word sense disambiguation. Therefore, it is weaker. This weakness could be diminished by using general word sense disambiguation relying on the context of the word in question as a future addition.

IV. Evaluation

As the goal of the project is to be able to import a large amount of data at once, the only tangible performance metric is whether the data extracted can be imported into Scone without triggering errors, as Scone checks for inconsistencies while loading new knowledge in. Unfortunately, while it is clear that individual cases do load properly, we are still waiting on the server to finish processing all of the data, since WordNet contains a lot of it.

V. Surprises and Lessons Learned

The main surprise was the relative ease of the presented solution, or rather the basic level of the concepts used. The work described previously hinges on using a parser for WordNet glosses that contains within it a higher-level ontology. Using dependency parsing seems to be a much more basic approach, and yet it is possible to identify some useful constructs from it alone. A related surprise was the fact that WordNet glosses are constrained enough to contain structures that can be identified and extracted, though in hindsight that is probably to be expected. The initial exploration of dependencies that might be useful was done by exploring the available dependencies and their typical appearance in sentences, so it was a bit relieving to see that those more theoretical descriptions match what is found in definition sentences.

VI. Conclusions and Future Work

In conclusion, this research project introduces an approach to extracting common-sense information from definitions by using dependency parsing and indentifying relevant structures that can be transplanted and used in Scone. It produces both a large import of knowledge and scripts that can be used to either regenerate that knowledge or identify similar structures in non-WordNet definitions, though still outputting Scone forms (both are made available at this project's website). If such knowledge is useful to another system that has similar expressive power, it would not be too hard to modify the output forms to reference that system instead.

The current approach can be improved and extended in a number of ways. Indentification of the patterns relies on some assumptions that may not

be true and thus might produce incorrect output. As stated before, the head of the definition may not truly be the correct word to use as the parent concept for the WordNet synset being processed. For example, if X is defined as a ”type of Y” rather than ”Y,” the head of that definition would be ”type” rather than ”Y,” which would thus be incorrectly identified as the parent concept of X. There are 69 such glosses in WordNet out of the 82,221 noun glosses overall. An improvement would be more careful about such edge cases.

Another flaw is ignoring negation in definitions. While it is not common due to the structure of definitions, negation completely changes the meaning of a sentence, meaning some incorrect information can be extracted if it is ignored. An improvement to the current approach would need to take such cases into account.

As mentioned in the roles section, the pattern used for roles is not necessarily specific enough to phrases that represent role-like information, nor does it cover all possible patterns. An investigation into how the incorrect cases could be identified and prevented would improve the first part of the problem. An investigation into other ways role-like information can be expressed would allow for more information to be extracted. One example case is the preposition ”from”: WordNet defines a plant process as a ”... projection ... from a plant body,” which means that a plant process is a projection on a body (this phrase could also be written as ”projection of a plant body” or ”projection on a plant body”) and should be identified as a role. However, since only ”of” prepositions are included right now, it will be ignored.

More substantial improvements could be done on the variety of patterns and the word sense disambiguation process. There are undoubtedly more patterns in dependency graphs whose corresponding structure in Scone could be identified. The current patterns could be also extended to become more complex: for example, including conjunctions would be a way to extract more information as in some cases definitions include two words as the ”parent concept”; those two words are generally joined with a conjunction.

Word sense disambiguation could use better heuristics when specifically WordNet glosses are concerned; it is likely that there are other ways to restrict a word sense to one of the possible ones. Both WordNet glosses and general definitions could benefit from more general word sense disambiguation, for example, focused on the context that the word in question appears in within the definition and compared with the usual contexts of the different senses of this word.

In short, while the project provided an initial approach and implementation, it could use future work that builds on the current progress.

ACKNOWLEDGMENTS

The authors would like to thank my advisor, Dr. Scott Fahlman, for his guidance during this project.

REFERENCES

[1] Allen, J., De Beaumont, W., Blaylock, N., Ferguson, G., & Orfan, J. (2011). Acquiring commonsense knowledge for a cognitive agent.

[2] Bird, S., Loper, E. & Klein, E. (2009). Natural Language Processing with Python. O'Reilly Media Inc.

[3] Chen, D., & Manning, C. D. (2014, October). A Fast and Accurate Dependency Parser using Neural Networks. In *EMNLP* (pp. 740-750).

[4] De Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006, May). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC* (Vol. 6, No. 2006, pp. 449-454).

[5] Fahlman, S. (n.d.). Scone User's Guide [PDF].

[6] Jurafsky, D., & Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, N.J, NJ: Pearson Education.

[7] Miller, G. A. (1995). WordNet: A Lexical Database for English. Communications of the ACM, 38(11), 39-41. doi:10.1145/219717.219748

[8] OWL Web Ontology Language Guide. (n.d.). Retrieved May 04, 2016, from https://www.w3.org/TR/owl-guide/

[9] Princeton University "About WordNet." (2010). WordNet. Retrieved May 04, 2016, from http://wordnet.princeton.edu

[10] Universal Dependencies. (n.d.). Retrieved May 04, 2016, from http://universaldependencies.org/introduction.html