1 Knowledge Extraction

One of the tasks of groups in Souma is the collection and exchange of knowledge. This knowledge is encoded in text postings since the use of language is the natural modus operandi for people when collaborating and operating on knowledge. Two problems arise from this:

1. Postings are used for two distinct and at times contradicting tasks: (1) Sharing knowledge with other users and (2) Storing knowledge in the system. For the former old postings should be steadily replaced with new ones. For the latter prominent display-space should be occupied with important posts rather than new ones.

2. The semantic processing of language is difficult for machines due to inherent lexical and grammatical ambiguities as well as to the context-dependent nature of language. This leads to a virtual inability of Souma to support users with knowledge-related tasks beyond the scope of simple storage; this includes semantic (rather than lexical) retrieval of knowledge, comparison of knowledge between groups and (machine-aided) inferencing.

To mitigate these problems a system for extracting knowledge from group-postings might be helpful. To asses the feasibility of such an approach a prototypical-implementation was carried out. The ensuing sections give an introduction to the field of knowledge extraction and representation, describe the implemented approaches and examine potential applications for Souma.

1.1 Introduction

Knowledge representation in computer science, especially in the field of Artificial Intelligence, is usually performed using a formalism called ontology [Guarino, 1995]. "An ontology describes a hierarchy of concepts related by subsumption relationships; [...] Other relationships between concepts [are also included]" [Guarino, 1998]. It is sensible to adhere to ontologies as knowledge representation formalism in the present case since ontologies and reasoning on ontologies are well-researched fields.

The task of extracting ontologies from text has gained increased attention lately. The usual approach consists of several extraction tasks building on each other and is organised in the so called *ontology learning layer cake* (see fig. 1) [Cimiano et al., 2009]. Several implementations exist as surveyed by Wong [Wong et al., 2012]. However their sources are either not available or they are not suited for an integration into Souma.

This led to the decision to attempt a prototypical implementation of `knoex`[1], an open-source Python tool for Ontology-extraction from text. The implementation is guided by techniques described in the aforementioned survey-papers, departing from them where necessary to avoid an undue dependency on data-mining techniques which would not yield adequate results given the limited

[1]`www.github.com/cartisan/knoex`

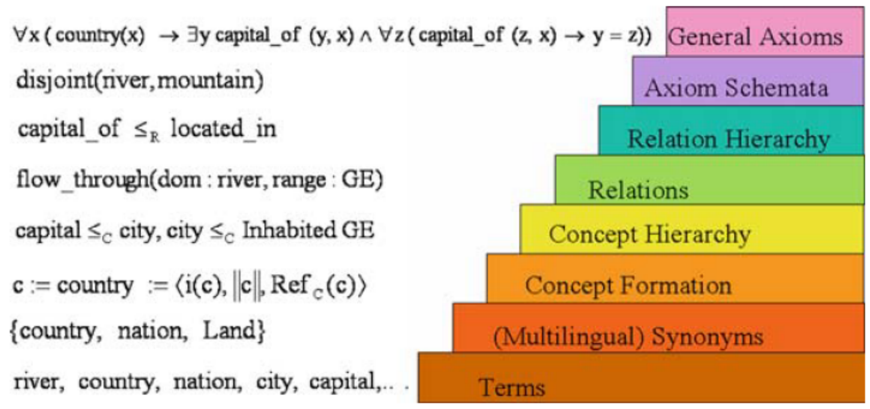
∀x (country(x) → ∃y capital_of (y, x) ∧ ∀z (capital_of (z, x) → y = z))
General Axioms
disjoint(river, mountain)
Axiom Schemata
capital_of ≤R located_in
Relation Hierarchy
flow_through(dom : river, range : GE)
Relations
capital ≤C city, city ≤C Inhabited GE
Concept Hierarchy
c := country := ⟨i(c), ‖c‖, RefC(c)⟩
Concept Formation
{country, nation, Land}
(Multilingual) Synonyms
river, country, nation, city, capital,...
Terms

Figure 1: Extraction tasks for ontology learning

length of the input data consisting of group-postings. The use-case for the prototype is to take an arbitrary text as input and extract a concept hierarchy from it, as well as the relations between those concepts.

1.2 Term Extraction

The first extraction task is to find all relevant terms in the text. A naive solution would be to perform part-of-speech tagging and extract all nouns. However since "English [...] make[s] extensive use of compounding" [Hippisley et al., 2005] terms often consist of multiple words like e.g. *contact lenses*. To capture such multi-word terms an implementation of the statistical C-NC measure according to Frantzi et al. [Frantzi and Ananiadou, 1997] was conducted.

First, part-of-speech tagging is performed[2]. Multi-word term candidates are then located using the Linguistic Filter provided by Frantzi: $(Noun|Adjective)^{+}Noun$. For each candidate term the according C-value is calculated.

$$\text{C-value'}(a) = \begin{cases} \log_2 |a| \cdot f(a) & \text{iff } |a| = \max \\ \log_2 |a| \cdot (f(a) - \frac{1}{c(a)} \sum_{i=1}^{c(a)} f(b_1)) & \text{otherwise} \end{cases} \qquad (1)$$

a	term-candidate
$\|a\|$	number of words in a
$f(a)$	frequency of a in the corpus
b_i	extracted term-candidates that contain a
$c(a)$	number of term-candidates that contain a

[2]Most of the language processing in `knoex` is done using Python's NLTK framework. However for POS-tagging the TextBlob library `http://textblob.readthedocs.org/en/dev/` was used due to its significantly better performance.

The C-value represents the *term-hood* of a term. It discounts the total frequency of a multi-word term if it shows insufficient independence from a containing multi-word term. For instance if *soft contact lens* is a candidate term, it entails the candidates *contact lens* and *soft contact*. Frantzi claims that equation 1 captures the independence of *contact lens* in the corpus and yields a high C-value. *Soft contact* however is supposed to be shown to be dependent on its containing multi-word and thus results in a low value.

The list of term-candidates is ordered according to their C-value and the candidates with the top-value are extracted. For each of these candidates the context in a window of size 1 (its left and right neighbors) is extracted. If the context word are either a verb, a noun or an adjective it gets assigned the following weight:

$$weight(w) = 0.5 \cdot (\frac{t(w)}{n} + \frac{ft(w)}{f(w)}) \tag{2}$$

w	context-word
n	total number of candidate terms
$t(w)$	number of distinct candidate terms that have w in their context
$ft(w)$	number of times w appears with candidate terms
$f(w)$	number of times w appears in the corpus

The list of candidate-terms is reranked according to the weight of its context. For each candidate term its context weight is calculated by summing over the *weight* of its context-words. If a context-word was not extracted during the last step its weight is considered to be zero.

$$wei(a) = \sum_{b \in C_a} weight(b) + 1 \tag{3}$$

a	term-candidate
C_a	context of a

A reranking according to the NC-value will now take into consideration both, the term-hood of a candidate as well as its context:

$$\text{NC-value}(a) = \frac{1}{\log N} \cdot \text{C-value'(a)} \cdot wei(a) \tag{4}$$

a	term-candidate
N	size of the corpus

The NC-value provides a ranking for term candidates. Its absolute value doesn't have much significance. Thus no threshold can be defined to filter out the terms. Instead a fixed percentage of the top candidates is extracted as terms. In the case of knoex this percentage has been set to 10%. However no systematic evaluation of this value has been performed yet.

1.3 Concept Formation

The next task is the formation of a concept hierarchy from the extracted terms. However to find the right concept for a term, disambiguation has to be performed. This is done using statistical methods and leveraging *WordNet* [Fellbaum, 1998] as a background-knowledge base.

WordNet is a taxonomy that organises terms into sets of synonyms called synsets. It is considered one of the biggest machine-readable lexical-semantic knowledge bases. NLTK provides access to the WordNet API, which allows to find all possible synsets for a term and to calculate the taxonomic similarity of two synsets.

Disambiguation in knoex rests on two assumptions:

1. Semantically close concepts are more likely to appear together than semantically unrelated concepts.
2. Semantical closeness is inversely correlated with taxonomic distance.

The first assumption can be supported by the consideration that a text is usually dealing with a topic. Thus most concepts in the text will be related to the topic and hence transitively to each other.

Building on this assumption the disambiguation of a set of n terms can be performed by creating the cartesian product of the synsets of all terms $\prod_{i=1}^{n} S_{t_i}$, and computing the overall semantical closeness osc for each n-tuple x in the cartesian product. Osc can be approximated by relying on the second assumption and employing WordNet's path similarity measure (which is inversely related to taxonomic distance). In this case $osc(x)$ is the sum of the path-similarities of all pairwise synset-combinations C^2 in an n-tuple of synsets x. Thus, the tuple $syns$ with the most likely combination of synsets can be obtained by maximizing overall semantical closeness.

$$syns = argmax_x(osc(x))$$

$$\text{with } x \in \prod_{i=1}^{n} S_{t_i}$$

$$\text{and } osc(x) = \sum_{(s_i,s_k)\in C^2} ps(s_i, s_k) \text{ with } C^2 = \{K | K \in \mathcal{P}(set(x)) \wedge |K| = 2\} \quad (5)$$

n	the number of terms to be disambiguated
$syns$	n-tuple of most probable synsets after disambiguation
$osc(x)$	Function computing the overall semantic closeness of an n-tuple of synsets x
S_t	set of all possible synsets for term t
$ps(s_1, s_2)$	WordNet path similarity of synset s_1 and s_2, (inversely) based on taxonomic distance, can take values between 1 (identity) and 0.

This seems to be a legitimate heuristic since it is true that a decreasing taxonomic distance leads to an increasing semantic closeness. However it remains only an approximation since semantic closeness can result not only due to taxonomic relatedness but also for instance due to associative relatedness[3].

The approach outlined above leads to a combinatorial explosion since for all possible synsets of a new term the distance has to be computed to all possible synsets of the other terms. In order to account for this a bootstrapping approach is implemented in knoex. In a first step the terms with a minimal number of potential synsets are identified. This subset of terms is disambiguated using the approach presented in equation 5. All other terms t are iteratively disambiguated in a second step. This happens by finding the synset c_t that maximazes the cumulative distance cd to all elements in the set of previously identified synsets S_{id}.

$$c_t = argmax_x(cd(x)) \qquad \text{with } cd(x) = \sum_{s \in S_{id}} ps(x, s) \text{ and } x \in S_t \tag{6}$$

t	term
c_t	the concept a term represents (after disambiguation)
S_t	set of all possible synsets for term t
S_{id}	set of already identified synsets
$cd(x)$	function computing the cumulative distance of a synset x to already identified synsets
$ps(s_1, s_2)$	WordNet path similarity of synset s_1 and s_2

This significantly reduces the number of distance computations. However the result of the disambiguation can be skewed by the choice of terms in step 1.

1.4 Relation Extraction

The extracted concepts can bear relations to each other. To extract this relations knoex resorts to a grammatical pattern matching approach that was inspired

[3]Thus for instance it could be claimed that *house cat* and *dog* are semantically more related concepts than *house cat* and *hyena*. Although both the house cat and the hyena are Felidae and thus taxonomically close, the connection of cat and dog as domestic animals is arguably more important.

by the usage of Hearst-patterns to extract hyponymy-relations [Hearst, 1992]. Hearst proposed a number of lexico-syntactic patterns like "NP_0 such as NP_1, $NP_2 \ldots$, (and|or) NP_n" that convey a hyponymy-relation between the participating NPs.

In the current state knoex leverages only one pattern of the form NP_0 VBZ NP_1. This pattern captures the subject/object (and respectively agent/patient) relation conveyed by verbs to extract that NP_0 stands in the relation of VBZ to NP_1. For instance applying this pattern to the phrase *"Knoex extracts information"* would yield the relationship *extracts* directed from a concepts *knoex* to a concept *information*.

To account for the nested nature of language, patterns in knoex are not simply applied on a part-of-speech tagged version of the text. Instead a constituency parse tree is computed using the Stanford Parser[4]. Using this tree all possible sequences of non-terminals representing the complete sentence are created, and the patterns are matched on these sequences.

The parse tree of an exemplar sentence, *"France is a nice country"*, can be seen in fig. 2. This sentence can be represented by the non-terminal sequences: *NNP, VBZ, DT, JJ, NN* or *NNP, VBZ,* NP_1 or NP_0, *VBZ,* NP_1; . . . . Applying pattern matching on these sequences leads to the extraction of the relation *is* between the concepts *France* and *a nice country*.

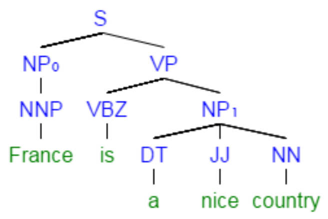

Figure 2: Exemplar parse tree produced by the Stanford Parser for the sentence: "France is a nice country."

After extracting the relations it is checked whether the terms they combine have already been disambiguated. If this is not the case they are disambiguated using step two of the bootstrapping procedure outlined in section 1.3 and added to the general concept pool.

1.5 Conclusion

The conclusion of the work on a prototypical implementation of a system for information extraction from text is three-fold. First of all a working proof of concept with limited capabilities was implemented. This leads to second: A number of insights on the difficulties of evaluating ontology extraction tools were identified and an assessment of conceptual shortcomings was conducted. Apart from this, third, a variety of missing functionality has been identified along with potential solutions on their realisation. The following sections deal with each of these topics.

[4]`http://nlp.stanford.edu/software/lex-parser.shtml`

1.5.1 Current State

In it's current state knoex is a working prototypical ontology extraction tool. It takes a natural language text as input and produces a collection of concepts, which are organised in a taxonomy, as well as relations that hold between those concepts. For the most general extracted concepts the ontology also contains their super-concepts in order to provide for better overall integration through background knowledge.

The resulting ontology is represented in an internal object-oriented format and can be displayed using simple graphical interface (example see fig. 3). No export into common formats like RDF or OWL is available yet.

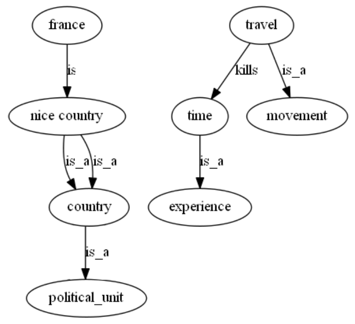

Figure 3: Graphical representation produced by knoex for the ontology extracted from the text: "France is a nice country. Travel kills time."

1.5.2 Evaluation

The used algorithms in many cases provide only basic functionality and could be either enhanced or exchanged. However before expanding on the prototypical implementation a thorough metric based evaluation should take place in order to allow for a performance comparison of different implementations.

Evaluating ontology extraction tools is a notoriously hard problem. No gold standards exist so far and a few trials have shown that even for humans it is hard to agree on an ontology for already slightly complicated texts. Thus it seems promising to evaluate the individual processing steps. A discussion of possible metrics and approaches can be found in Wong et al. [Wong et al., 2012].

Even before performing a metric based evaluation a few shortcomings of the current implementation can be identified. The NC-value based extraction of terms relies on a fixed percentage of top candidates to be extracted. Preliminary inspection has shown it to perform rather unsatisfactory due to difficult parameter tuning. Either the false-negative or the false positive rate is too high. One reason might be the rather arbitrary choice of the linguistic filter. Considering special patterns of part-of-speech tags without taking into account the dependencies of the constituents does not seem cognitively adequate. A solution might be to change the linguistic filter to select noun phrases after performing a dependency parsing of the sentences. Since (at least kind-referring) NPs exhibit similar behaviour to multi-word terms the NC-value could also be applied to them[5].

[5]The two linguistic properties that are relevant for the application of the NC-value are nestedness and the head-noun principle. Nestedness is relevant because the NC value is

The disambiguation process relies on the assumption that semantic closeness inversely correlates with taxonomic distance. This seems to be a legitimate heuristic since it is true that a decreasing taxonomic distance leads to an increasing semantic closeness. However since it can not be stated that a concept c_3 is semantically closer to a concept c_1 than to a c_2 iff $\delta_{tax}(c_1, c_3) < \delta_{tax}(c_2, c_3)$ this remains only a heuristic that has to be evaluated empirically.
The extraction of relations is based on lexico-syntactic patterns. Entering these patterns by hand is a tedious and unsatisfactory task.

1.5.3 Future Work

Apart from improving the performance of the current implementation it also has to be expanded to deal with more complicated problems in order to increase the expressibility of the obtained model.

Most of the ontology extraction literature as well as knoex deals with the extraction of T-box knowledge. However it is a trivial observation that natural language deals not only with describing concepts but also characterising individuals. Thus the extraction of A-box knowledge is also important to capture the knowledge represented in a text. A first approach to extract instances would be to perform a Named Entity Recognition task. However not all instances are represented as named entities. Also definites represent uniquely identifiable entities in the discourse (e.g. “The walkie-talkie was old and battered”). This holds with the exception of weak definites that can take a kind-referring as well as individual-referring meaning (e.g. “The hospital is where you go to get healthy”). The distinction between weak definites and strong definites as well as the different interpretations of weak definites is subject to an ongoing linguistic debate (e.g. [Aguilar-Guevara and Zwarts, 2013]).

An additional problem with A-box is the attachment of extracted instances to known concepts. This often requires common sense knowledge (as in the case of e.g. “The Hitchhiker’s Guide to the Galaxy”) or an analysis of context (e.g. “*Arthur Dent* was an old bachelor“ where Arthur should be recognized as a human (or at least a British bloke) as opposed to e.g. a dog).

This problems impede state of the art systems from extracting A-box knowledge and make further research necessary.

Another important task is the extraction of properties. While knoex is capable of extracting multi-term concepts like e.g. ”green snake“ as well as the super-concept ”snake“, it is not capable of extracting the difference between the

directed at finding the right level of granularity for a nested combination of terms. The head-noun principle makes sure that the extracted multi-word structure is modifying one particular structure (i.e. a noun). This provides a justification to extract the multiword-structure as one term. According to [Krifka, 1995] at least kind-referring NP always characterise the head-noun. Non-kind-referring NPs might be taken as relating to instances rather than concepts - a topic that will be discussed later.

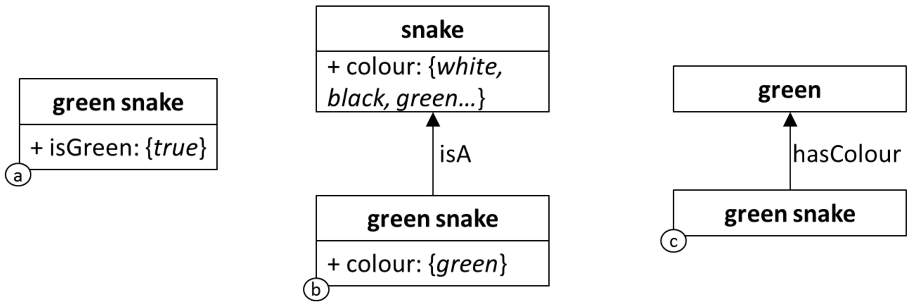

Figure 4: Three possibilities to encode the concept “green snake” in an ontology. Concepts are written in bold face, property names prefixed with a “+” and property domains are denoted in set notation. Relations are represented using directed edges.

two. Thus the semantic load is shifted to the user who is to infer that the first is probably a specialisation of the latter due to the color being green.

There seem to be three possible ways to represent the aforementioned concept (see fig. 4). A first solution (a) would be to use a dependency parser to obtain the modifying constituents for each concept and use them as boolean properties with the range fixed to the value of *true*. For instance ”green snake“ would have ”isGreen“ extracted as property with a range of *true*.

While this is a first step in distinguishing sub-concepts it doesn’t leverage the inheritance principle of taxonomies well. Thus an alternative (b) would be to use background knowledge data to infer the type of the extracted property and use this in turn as a property in the super-concept. This would result in ”green snake“ having a fixed-range property of ”colour: *green*“ and in ”snake“ having a property ”colour“. It is however not clear how far up the hierarchy such a propagation of properties could go.

A third alternative (c) would be to use relations for the realization of properties. This would mean to create concepts for extracted properties and connect the modified concept with the modifying concept via an appropriate relation (inferred from background knowledge). In that case ”green snake“ would have a relation ”hasColor“ with the concept ”green“. This solution would deal with the rather inelegant necessity to restrict the range of properties as outlined above. However it would result in a significant bloat of the concept space.

Which ever solution is the most suitable has to be evaluated empirically. Apart from the first one however all of them need to infer the type of properties which is a non-trivial task (even for humans it is hard to infer ad-hoc the type of a modifier - for instance what would be the type of “poisonous” as in “poisonous green snake”?). A remedy could be found in the form of generative lexica like the Brandeis ontology (cf. [Pustejovsky et al., 2006]) that help generating the semantics of a word from it’s position in an ontology.

Further investigation has to be done on classical topics like axiom extraction and anaphora resolution.

1.6 Potential Application for Souma

Three potential use cases for an ontology extraction tool in the context of a social network have been identified so far. Since knoex and Souma have not been combined yet this remains a theoretical analysis, that however is informed by the results obtained so far from the implemented prototype.

The most obvious use is the separation of knowledge and text that knoex can provide. As described in section 1 postings in groups are used for two contradicting tasks: Sharing knowledge with others and storing knowledge for later use. The former requires postings to be displayed by recency while the latter requires postings to be displayed by importance. Since knoex can extract the knowledge from a posting and store it an ontology the posting has only to fulfill it's sharing task. The storage can ensue in the ontology.

Extracting knowledge from a group thus changes from searching for old posts to querying a knowledge base. This is an enhancement since text search suffers from natural language ambiguities while querying a knowledge base operates on a semantic rather than on a linguistic level.

The second use case is the comparison of group knowledge. If a knoex instance would be set up for every group it could extract knowledge from the group postings into the group's personal ontology. By comparing the ontologies of different groups potential new collaborators across group boundaries could be identified as well as new topics that might also interest a particular group. To avoid miscommunication between groups their usage of concepts could be compared. This can happen by extracting the closure for concepts (i.e. all concepts that can be reached via one relation) in question from the respective ontologies and comparing the different associations.

A last use case is the automated tagging of posts that might be required for creating an interest model. Potential hashtag-candidates for a post are the terms extracted from it. These however suffer from natural language ambiguities. A disambiguation can happen by finding the concepts of a post and using an appropriate super-concept as hashtag. This would constitute a step of abstraction that is helpful in tagging as e.g. a post mentioning "hunter terriers", "golden retrievers" and "collies" would be appropriately tagged by the generalisation "dog". It of course requires further research in order to not end up tagging all posts with a semantically vacuous top-concept, this however can most likely be solved through appropriate heuristics.

A possible solution would be to use the similarity measures of the background knowledge-base in a manner comparable to the one described in section 1.3 to identify the central concepts of a post. This can than be used to both, produce simple tags as well as serve as starting points for an abstraction process.

Tags generated in this manner can successively be used to implement parts of the interest model. As described in section ?? Souma uses a Latent Dirichlet Allocation (LDA) model trained on a dump of Wikipedia to identify the topics of a document. This topics are used as features in a Naive Bayes (NB) model of each user for a binary classification of posts in *likes* and *dislikes*. However as discussed in section ?? the trained LDA model falls short of creating informative topics. It can be argued that the central concepts as identified by the approach indicated above would fulfill the same task as the LDA topics. Thus the NB classifier could be trained on the set of WordNet synsets as features. Tagging posts would than provide the features of an individual post that NB needs for classification. This would allow to implement the interest model without the problematic LDA part.

Acknowledgement

The theoretical and practical work on knoex has been performed in the interdisciplinary course "Developing a cognitive social Network". We would like to thank Moritz Meier and Kai Standvoss for their work and PD Dr.Helmar Gust for the active supervision.

References

[Aguilar-Guevara and Zwarts, 2013] Aguilar-Guevara, A. and Zwarts, J. (2013). Weak definites refer to kinds. *Recherches linguistiques de Vincennes*, 42:33–60.

[Cimiano et al., 2009] Cimiano, P., Mädche, A., Staab, S., and Völker, J. (2009). *Ontology Learning*, pages 245–267. Handbook on Ontologies. Springer Verlag.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

[Frantzi and Ananiadou, 1997] Frantzi, K. and Ananiadou, S. (1997). Automatic term recognition using contextual cues. In *Proceedings of 3rd DELOS Workshop*.

[Guarino, 1995] Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *Int. J. Hum.-Comput. Stud.*, 43(5-6):625–640.

[Guarino, 1998] Guarino, N. (1998). Formal ontology and information systems. pages 3–15. IOS Press.

[Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora.

[Hippisley et al., 2005] Hippisley, A., Cheng, D., and Ahmad, K. (2005). The head-modifier principle and multilingual term extraction. *Natural Language Engineering*, 11.

[Krifka, 1995] Krifka, M. (1995). *The Generic Book*, chapter Genericity: An Introduction. The University of Chicago Press.

[Pustejovsky et al., 2006] Pustejovsky, J., Havasi, C., Littman, J., Rumshisky, A., and Verhagen, M. (2006). Towards a generative lexical resource: The brandeis semantic ontology. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA).

[Wong et al., 2012] Wong, W., Liu, W., and Bennamoun, M. (2012). Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20:1–20:36.