WebChild: Harvesting and Organizing Commonsense Knowledge from the Web

Niket Tandon
Max Planck Institute for Informatics
Saarbrücken, Germany
ntandon@mpi-inf.mpg.de

Gerard de Melo*
IIIS,Tsinghua University
Beijing, China
demelo@tsinghua.edu.cn

Fabian Suchanek
Télécom ParisTech
Paris, France
fabian.suchanek@telecom-paristech.fr

Gerhard Weikum
Max Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

This paper presents a method for automatically constructing a large commonsense knowledge base, called WebChild[1], from Web contents. WebChild contains triples that connect nouns with adjectives via fine-grained relations like `hasShape`, `hasTaste`, `evokesEmotion`, etc. The arguments of these assertions, nouns and adjectives, are disambiguated by mapping them onto their proper WordNet senses. Our method is based on semi-supervised Label Propagation over graphs of noisy candidate assertions. We automatically derive seeds from WordNet and by pattern matching from Web text collections. The Label Propagation algorithm provides us with domain sets and range sets for 19 different relations, and with confidence-ranked assertions between WordNet senses. Large-scale experiments demonstrate the high accuracy (more than 80 percent) and coverage (more than four million fine grained disambiguated assertions) of WebChild.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Knowledge Bases; Commonsense Knowledge; Web Mining; Label Propagation; Word Sense Disambiguation

1. INTRODUCTION

Motivation. Automatically constructed knowledge bases (KBs) have recently proven to be great assets for Web search, recommendations in social media, and text analytics. Prominent examples are the Google Knowledge Graph and the (small but effective) use of KBs in the IBM Watson project for deep question answering [29, 18]. The largest publicly available KBs are `dbpedia.org`, `freebase.com` and `yago-knowledge.org`. The strength of these KBs is in taxonomic and factual knowledge: entities in semantic classes and relationships between entities. However, they are fairly ignorant regarding commonsense knowledge: properties and statements that are easily picked up by children, but are hard to acquire by a machine.

For example, all humans know that apples are round and carrots are orange and have a longish shape. Computers completely lack this kind of commonsense knowledge, yet they would enormously benefit from such an asset for various use-cases of growing relevance: language understanding for translation or summarization, speech-based human-computer dialog, faceted search and search query suggestions, sentiment analytics on social media, and more. This paper presents a methodology for automatically extracting and cleaning commonsense properties from the Web. The resulting KB is called *WebChild*.

State of the Art and its Limitations. Prior work on commonsense knowledge includes the seminal projects Cyc [21] and WordNet [12], the more recent work on ConceptNet [31], and the work by [35] and [20]. Cyc has compiled complex assertions such as *every human has exactly one father and exactly one mother*, but did not aim to gather properties of things at large scale. WordNet has manually organized nouns and adjectives into lexical classes, with careful distinction between words and word senses; however, nouns and adjectives are not connected by any semantic relation, except the extremely sparse `attribute` relation (with around 1,200 links). ConceptNet is a huge collection of commonsense assertions, but the vast majority are instances of generic relations like `IsA`, `PartOf`, `ConceptuallyRelatedTo`, or `DerivedFrom`. The more specific relations like `adjectivePertainsTo` or `UsedFor` have only few instances. Tandon et al. [35] automatically compiled millions of triples of the form ⟨ *noun relation adjective* ⟩ by mining n-gram corpora, but the relations are still fairly generic such as `HasA`, `HasProperty`, or `CapableOf`. Lebani & Pianta [20] proposed encoding additional lexical relations for commonsense knowledge into WordNet, but their approach is inherently limited by relying on human input and also focuses on simple relations such as `usedFor`, `partOf`, etc.

None of these knowledge resources have refined properties like shape, size, taste, emotion, etc., and none have pro-

[1]WebChild: datasets will be publicly available at `http://mpi-inf.mpg.de/yago-naga/webchild/`

*Gerard de Melo's work was supported in part by the National Basic Research Program of China Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WSDM'14, February 24–28, 2014, New York, New York, USA.
Copyright 2014 ACM 978-1-4503-2351-2/14/02 ...$15.00.
http://dx.doi.org/10.1145/2556195.2556245.

duced large amounts of semantically disambiguated knowledge that distinguishes the different meanings of ambiguous properties such as *hot*, which can refer to temperature, taste, or emotion. For example, the KB by [35] would merely have simple triples like ⟨*milk*, hasProperty, *hot*⟩, ⟨*chiliPepper*, hasProperty, *hot*⟩, ⟨*dress*, hasProperty, *hot*⟩. Thus, state-of-the-art commonsense KBs still have severe limitations: i) sparseness on aspects that go beyond generic relations, ii) focus on crude relations, without distinguishing different semantic properties, and iii) no distinction between words and their different senses.

Our Approach. This paper presents WebChild, a large commonsense KB automatically built from Web sources by a novel method relying on semi-supervised learning. WebChild contains more than 4 million triples for fine-grained relations such as hasTaste, hasShape, evokesEmotion, etc. We use a judiciously designed form of *label propagation (LP)* (see [34] for an intro) for learning the domain set, the range set, and the extension of such relations, at large scale. To this end, we first construct graphs that connect nouns, adjectives, and WordNet senses as nodes, by weighted edges. The edge-weights are derived from sense relatedness, pattern statistics, and co-occurrence statistics. We harness WordNet and Web data to obtain seeds to initialize the LP graphs, and then use LP algorithms to derive high-quality assertions for fine-grained relations between noun senses and adjective senses.

Contributions. Our methodology has a number of salient characteristics and results in a large commonsense KB with unique qualities:

1 **Fine-grained assertions:** WebChild is the first commonsense KB that provides refined hasProperty relationships between nouns and adjectives into specific and thus more informative relations. We support 19 different relations like hasShape, hasSize, hasTaste, evokesEmotion, etc.

2 **Disambiguated arguments:** The arguments of all assertions in WebChild are disambiguated by mappings to WordNet senses: noun senses for the left-hand arguments of a relation, and adjective senses for the right-hand arguments.

3 **Minimal supervision:** Our method does not require any labeled assertions for training. Instead we use bootstrapping based on Web patterns and WordNet. Our method copes well with noisy input data.

2. MODEL AND METHODOLOGY

2.1 Goal and Notation

We aim to compile a large and clean set of fine-grained commonsense properties, connecting noun senses with adjective senses by a variety of relations. In contrast to prior work that only dealt with a generic hasProperty relation, we use 19 different (sub-)relations like hasShape, hasSize, hasTaste, hasAbility, evokesEmotion, etc. The list of relation types is derived from WordNet. A noun that is the target of WordNet's attribute relation (e.g., *shape*, *size*) becomes a relation in our knowledge base.

Our goal is to populate these relations with *assertions* in the form of triples $\langle ns, r, as \rangle$ where ns is a noun sense in WordNet, as is an adjective sense in WordNet, and r is one of the considered relations. Each relation r has a domain $dom(r)$, the set of noun senses that appear in r as left-hand arguments, and a range $rng(r)$, the set of adjective senses that appear in r as right-hand arguments.

It is important to distinguish word senses from words that occur in surface text, because nouns and adjectives are often highly ambiguous. For example, the adjective *green* is associated with both plants in the botanical sense and power plants. WebChild represents assertions for the different word senses in the following form:

⟨botanical-plant, hasColor, green-color⟩
⟨power-plant, hasQuality, green-environmental⟩

where botanical-plant, power-plant, green-color, and green-environmental are specific WordNet senses[2]

Throughout this paper, we carefully distinguish words from their senses, and use different notation and fonts to make this explicit. Surface words (or composite phrases) are written in italics, whereas senses appear in typewriter font and typically suffixed with either $-\mathtt{n}^i$ for noun senses or $-\mathtt{a}^j$ for adjective senses. Here, i and j are WordNet sense numbers if a noun or adjective has multiple senses (which is usually the case). For example, *chili* is a noun and *hot* is an adjective (both are words), whereas chili-n^2 and hot-a^9 are senses.

2.2 Sub-Tasks

We decompose the problem of finding assertions for fine-grained commonsense relations into three sub-tasks.

1. **Range Population:** First, we compute adjective senses that occur in the range of each of the WebChild relations. For example, for the hasColor relation, we obtain a list of color attributes including e.g. green-a^1, the color sense of *green* from WordNet, but not the environmental sense of *green*. For the hasShape relation, we obtain a list of possible shapes, e.g. circular-a^2.
2. **Domain Population:** Our second task is to compute noun senses for the domain of each relation. For example, war-n^1 for the evokesEmotion relation, and pizza-n^2 for the hasTaste relation.
3. **Computing Assertions:** Finally, we aim to map generic word-level assertion candidates ⟨*noun*, hasProperty, *adjective*⟩, gathered from Web corpora, into fine-grained assertions about word senses. For example, ⟨*car*, hasProperty, *sweet*⟩ is mapped into ⟨car-n^1, hasAppearance, sweet-a^2⟩.

2.3 Candidate Gathering

For all three sub-tasks we can start with a small number of *seeds* obtained from WordNet, for example, by using the *attribute* information that connects relational noun senses (e.g., shape-n^1) with adjective senses (e.g., straight-a^1 and crooked-a^1). This is very sparse data (e.g., there are only 2 adjective senses for the attribute shape). Our specific choice of seeds depends on which of the three sub-tasks we are dealing with. This will be discussed later in the respective sections.

To build a knowledge base of high coverage, we gather candidates for assertions from the Web. For this purpose, we harness a

[2]WordNet shows these senses as *plant (a living organism ...)*, *power plant (an electrical generating ...)*, *green (of the color between blue and yellow ...)*, and *green (concerned with ... Green Party)*.

huge *N-gram corpus*: the Google Web 1T N-Gram Dataset Version 1 [7], which consists of 1.2 billion 5-grams (i.e., 5 consecutive words or other tokens) derived from the index of the Google search engine. Each of these 5-grams comes with its frequency of occurrences on the Web. Thus, we can use these frequencies to simulate a full Web corpus. However, we also face the restriction that N-grams are limited in length to 5.

To gather assertion candidates from this data, we employ surface patterns whose matches return N-grams that contain a noun and an adjective that are likely to be related, in a generic `hasProperty` sense. Note that the resulting candidates are still at the word level; there is no way of mapping them to senses at this stage. We define generic templates for lexical patterns of the form

"<noun> *linking_verb* [*adverb*] <adj>" or
"<adj> <noun>".

Linking verbs are different forms of "to be", "to become", "to smell", "to taste", etc.[3] Our templates capture many variations of assertions. Examples are

apple was really <adj>,
apple was <adj>,
<adj> apple.

Applying this family of patterns to the Google N-gram corpus results in 3.6 million noun-adjective pairs. Many of these are noise (i.e., incorrect), and none of them is disambiguated onto senses yet.

2.4 Semi-Supervised Inference on Graphs

The candidates obtained by the outlined procedure are usually very noisy, not yet disambiguated, and not yet assigned to our fine-grained relations – they are just word pairs for the generic `hasProperty` relation and are still ambiguous. To distill the good pairs from the noisy pool and to map words onto proper senses and noun-adjective sense pairs into specific relations, we use a semi-supervised classification method over judiciously constructed graphs. To this end, we employ the method of *Label Propagation (LP)* [37].

For each of the three sub-tasks towards building WebChild, we construct a graph with words (or word pairs) and possible word senses (or sense pairs) as nodes. A small number of nodes encodes seeds, with known relation labels. Edges reflect the relatedness of nodes, with weights derived from Web statistics and WordNet information. The specifics of the graph depend on the sub-task that we are dealing with, and will be discussed in the following sections.

LP computes scores for nodes having certain labels. In our setting, these labels are used to distinguish different relation types. For inference, we use the *MAD (modified adsorption)* algorithm [34], which has shown good performance for graphs with high numbers of incident edges per node. Our graphs have this property because adjectives usually have many possible senses.

MAD propagates labels to neighboring nodes along the graph's edges; a high edge weight implies that the incident nodes are likely to have the same label. Seed nodes are expected to retain their original labels. Additionally, regularization is employed to minimize label changes within a neighborhood, which is essential to avoid overfitting. To encode this intuition, the MAD algorithm minimizes a loss function. Assume that the graph is represented as a weighted adjacency matrix W and that the label vectors of the nodes are encoded into matrices Y for the initial labeling and $\hat{Y}$ for the final predicted labeling. (Y_{*l}) and $(\hat{Y}_{*l})$ denote the l^{th} column vector of the initial matrix Y and the final label matrix $\hat{Y}$, respectively. Then the loss function is:

$$L(\hat{Y}) = \sum_l \Big[(Y_{*l} - \hat{Y}_{*l})^T S^l (Y_{*l} - \hat{Y}_{*l}) + \mu_2 \hat{Y}_{*l}^T L \hat{Y}_{*l} + \mu_3 \left\| \hat{Y}_{*l} - R_{*l} \right\|_2 \Big], \quad (1)$$

The first term encodes that initial and final labels for seed nodes should mostly be the same. This is enforced by the diagonal matrix S having S_{vv} = 0 for non-seed nodes, while for seed nodes S_{vv} is set to monotonically increase with the entropy of a node's transition probabilities (such that high degree nodes are discounted). The second term encodes that neighbor nodes obtain similar labels. This effect is realized by the unnormalized graph Laplacian L of the weighted adjacency matrix W. The third term contributes to the regularization of the estimated labels, in order to avoid over-fitting to their seed labels. This is enforced with an abandonment matrix R having a zero-valued column vector corresponding to every label, except the dummy label (the dummy label is an additional label that has a large value if the node cannot be properly labeled). A pre-defined weight is computed for the dummy label, in proportion to the nodes' degrees.

The MAD algorithm is a variant of the Jacobi method (also used for PageRank, for example), an iterative process that uses the current labels of nodes to update the label scores for neighboring nodes. When this process converges or a specified number of iterations is reached, each vertex is associated with a vector indicating the estimated labels (including the dummy label). The dummy label has a large value if the node cannot be properly labeled. In our setting, we apply this procedure for each relation separately, comparing the relation labels vs. the dummy label in the resulting output. We accept the relation label only if its final score is larger than that of the dummy label.

3. POPULATING RELATION RANGES

We now discuss how we are able to apply this same methodology to each of the three sub-tasks introduced in Section 2.2. Our first sub-task addresses the problem of identifying possible adjective senses for the range of each of the relations supported by WebChild. For example, for `hasTaste`, we expect adjectives like *delicious*, *spicy*, *hot*, *sweet*, etc., whereas these adjectives do not make much sense for the `hasShape` relation. The main difficulty that we face with this task is to move from the word level to word senses. So actually, we aim to populate the range of `hasTaste` with senses `delicious-a`[2], `spicy-a`[1], `hot-a`[9], `sweet-a`[1], etc. Some of these surface words also appear with other relations; for example, *hot* may also denote a property for the `hasAppearance` relation, however with different senses: `hot-a`[10] and `sweet-a`[4]. The task is to carefully discriminate the senses for the ranges of different relations (although some overlap between relations may be possible).

We solve this problem in three steps:

1. **Gathering candidates** from N-grams and other sources.
2. **Constructing a graph** that encodes association strengths between adjectives and adjective senses by weighted edges.

[3] see `http://en.wikipedia.org/wiki/List_of_English_copulae` for a full list

3. **Inferring adjective senses for a relation's range** by semi-supervised Label Propagation.

Candidate Gathering. We start with the raw candidates derived by extraction patterns from the Google N-gram corpus, as described in Section 2. For relation r, we filter the candidates by checking for the presence of the word *r*, any of its synonyms (e.g., *shape*, *form*, etc., or *appearance*, *look*, etc.), or any linking verb that is derivationally related to *r* (e.g., "tastes" for `hasTaste`). We apply these patterns also to WordNet glosses (i.e., short descriptions of word senses), to collect further candidates.

In addition, we apply the Hearst pattern "<r> such as <x>" to the N-gram data, and collect the matches for x as possible adjectives for relation r. Finally, we adopt the WebSets method [9] to HTML tables in specific articles of the English Wikipedia. The articles of choice are those whose article name corresponds to relation r (or its synonyms). These were manually identified for each relation r.

In total we collected around 40,000 adjectives for all relations together.

Graph Construction. So far we have merely compiled a large set of – mostly ambiguous – words that may populate the range of a relation. We use these words as nodes in a graph, and extend this node set by *all possible adjective senses* of these words. This is a simple lookup in WordNet, without any disambiguation yet.

Definition [Range Population Graph (RPG)]:
The RPG of a relation r is a weighted undirected graph with nodes V_{RPG} and edges E_{RPG} as follows:

- V_{RPG} consists of all candidate adjectives for relation r, and all their corresponding adjective senses.
- E_{RPG} consists of three kinds of edges:
  - edges between two words *w1* and *w2* if they share at least one noun in their pattern occurrences;
  - edges between two senses `w1-a`^{i} and `w2-a`^{j} if they are related in the WordNet structure or have WordNet glosses that suggest relatedness;
  - edges between a word *w* and all its senses `w-a`^{i}.

The left part of Figure 1 shows an example of an RPG.

Edge Weighting. To define meaningful edge weights, we utilize statistics from the candidate gathering (see Section 2) and from WordNet.

For weighting the *edges among words*, we harness the co-occurrences of adjectives with nouns. We derive from the large N-gram corpus two matrices $O : noun \times adjective$ and $P : noun \times adjective$ where O_{ij} is the number of occurrences of the noun-adjective pair and P_{ij} is the number of distinct extraction patterns that the noun-adjective pair occurs with. We normalize both matrices to have values in $[0, 1]$. For O, we divide all values by the maximum value. For P, we transform all values using the sigmoid function $f(x) = 1 - \frac{1}{e^{x-1}}$. The rationale here is to reward multiple patterns, but consider also the diminishing returns of observing many patterns. Finally, we combine O and P by the linear combination $\alpha O^T \times O + (1 - \alpha)P^T \times P$ with hyper-parameter α. The values of the resulting matrix are the weights for edges between two adjectives of the RPG.

For *edges between two senses* `u-a`^{i} and `w-a`^{j}, we consider their taxonomic relatedness within WordNet. If there is a path between `u-a`^{i} and `w-a`^{j} using `hypernym`/`hyponym`, `derivationally_related`, `similar_to`, `also_see`, or `antonym` links in WordNet, then we use the Hirst measure of semantic relatedness [17].

If no such path exists, we resort to the glosses of `u-a`^{i} and `w-a`^{j}, expanded by glosses of their respective hyponyms and hypernyms. We then compute the number of overlapping words shared by these contexts. This is essentially the concept similarity measure by [22]. All these measures are normalized to fall between 0 and 1, and we use a down-weighting coefficient for the gloss-based values, to ensure that path-related sense pairs have higher edge weights.

For *edges between words and senses*, we would ideally like to use the sense frequencies as a basis for edge weights. However, such information is hardly available.[4] We thus resort to the following statistics-based heuristics (adopted from [22, 25]). For each word *w*, we take the corresponding column from matrix O (i.e., the frequencies of co-occurring nouns) as a distributional-semantics vector. For each possible sense `w-a`^{i}, we compute a context vector from its gloss and the glosses of neighboring senses, giving us another noun distribution. The normalized scalar product between the vector of *w* and the vector of `w-a`^{i} is the weight of the edge between the adjective and its possible sense.

Label Propagation (LP). The final step is to run the MAD algorithm for Label Propagation on the constructed graph – one graph for each relation. We consider only two labels for each graph: the relation of interest and the dummy label (encoding *no relation* or *other relation*). We obtain seeds automatically by observing that the intersection of adjectives found in WordNet and on the Web, i.e. in more than one source, are more likely to be accurate. The sense of the WordNet adjective is considered for this. 30% of the remaining seeds were used as held-out test data to tune the parameters μ_2 and μ_3 of the MAD algorithm (see Section 2.4). The MAD algorithm then infers which adjective senses belong the range of the relation.

4. POPULATING RELATION DOMAINS

After populating the ranges of WebChild's relations, we turn to the relation domains. For each relation, such as `hasTaste`, we aim to compute the noun senses that can appear as left-hand arguments of the relation, for example, `apple-n`^{1}, `pizza-n`^{1}, `plant-n`^{2}, `beef-n`^{2}, but not `car-n`^{1}, `cow-n`^{2}, or a different sense of *plant*: `plant-n`^{1} (the industrial plant). Analogously to the previous section, we solve this task in a three-step process: gathering candidates, constructing a graph, and LP inference for cleaning and disambiguation. We will see that we can harness the knowledge that we already acquired about adjective senses that appear in relation ranges.

Candidate Gathering. We use the coarse-grained generic `hasProperty` noun-adjective pairs (n, a) gathered by the method of Section 2.3. Given a pair (n, a), if the adjective a has at least one sense that appears in the relation's range computed in Section 3, then the noun n becomes a domain candidate. For example, given word pair (*beef*, *salty*) and having the knowledge that *salty* occurs in the range of `hasTaste`, we infer that *beef* is a noun candidate for `hasTaste`.

[4]WordNet provides senses frequencies only for a small set of words, mostly nouns, and not nearly for all of their senses.

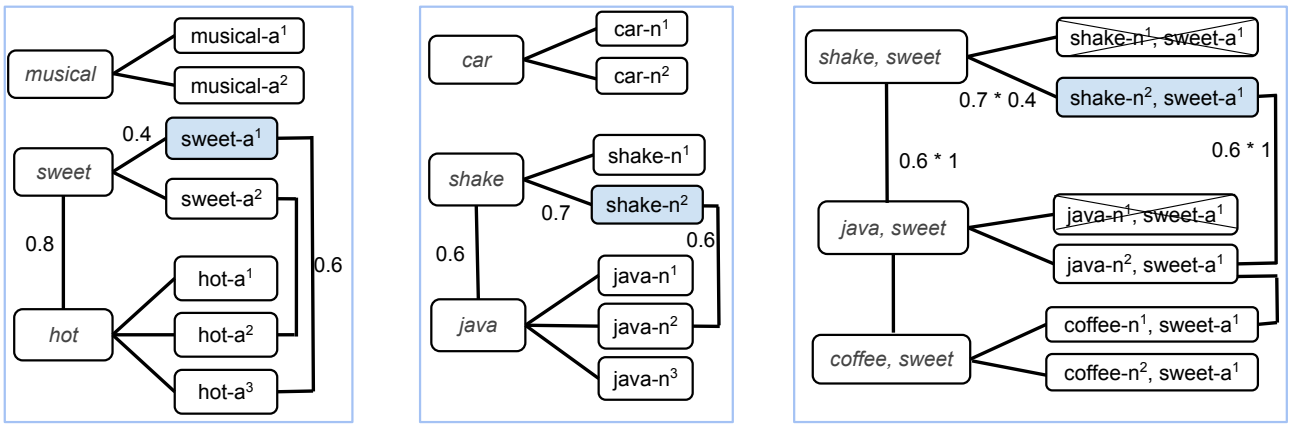


Figure 1: Graph construction for range (left), domain (middle), and, assertions (right). Blue nodes are seeds. Crossed nodes (right) denote assertion candidates pruned based on domain and range.

When n has only one sense in WordNet, we directly use this noun sense. However, this situation is rare. A more typical candidate would be *java*, with co-occurrence pairs such as (*java*,*tasty*), (*java*,*easy*), (*java*,*hilly*), etc. In such situations, to move from words to senses at least in some of the cases, we harness the glosses of adjectives in WordNet to derive *semi-disambiguated* assertions where either the noun or the adjective is mapped to a WordNet sense.

Gathering *semi-disambiguated* assertions: For each adjective word in our candidate set, we find all noun-sense glosses where the adjective occurs (as a surface word). Whenever a matching noun sense gloss is found, the specific noun sense is used to replace the ambiguous surface noun in the candidate pair. We perform this analogously for nouns in adjective-sense glosses. For instance, the gloss for `sour-a`^{2} reads "the taste experience when vinegar or lemon juice is taken ...". We generate two assertions from this: (*vinegar*, `sour-a`^{2}) and (*lemon juice*, `sour-a`^{2}). Note that although this technique goes further than the method for relation ranges, we still face a large amount of noisy candidates. An adjective such as *large* has seven word senses in WordNet, and we can obtain numerous noun-sense candidates whose glosses contain "large".

Graph Construction. Next, we construct a graph for subsequent Label Propagation similarly as in the method for range population (Section 3).

Definition [Domain Population Graph (DPG)]:
The DPG for a relation r is a weighted undirected graph with nodes V_{DPG} and edges E_{DPG} as follows:

- V_{DPG} consists of all candidate nouns for r and their possible noun senses from WordNet.
- E_{DPG} edges and their weights are computed exactly as described for RPGs (Section 3), with nouns taking the place of adjectives. The only difference is that we use a different sense similarity measure [19], because WordNet's nouns form a hierarchy and are thus very differently structured from the adjectives.

The middle part of Figure 1 shows a DPG example.

Label Propagation (LP). Again, we run the MAD algorithm for Label Propagation on the DPG for each relation. To generate seeds, we use `hasProperty` triples that have unambiguous nouns and adjectives that have been assigned to only a single relation r by the Range Population method. Some of these nouns are genuinely unambiguous while others are unambiguous for us because we have previously identified the correct sense using the WordNet gloss heuristics mentioned previously in *semi-disambiguated* assertion gathering. In either case, the single noun senses of such unambiguous nouns serve as seeds. The parameters of the MAD algorithm were tuned as described previously using held-out data. The MAD output provides us with scores for the noun senses of the ambiguous nouns. For the domain of relation r, we accept the noun senses whose score exceeds the dummy label score.

5. COMPUTING ASSERTIONS

Finally, we leverage the domain and range knowledge for distilling the raw and ambiguous assertion candidates for the generic `hasProperty` relation, gathered as explained in Section 2, into word-sense pairs for fine-grained relations. Thus, we need to disambiguate the left and right arguments of the candidate assertions and determine the respective relations. Again, we build a graph representation for each relation r, and apply Label Propagation on these graphs.

For a candidate assertion with an ambiguous noun and adjective, we would often generate a large set of nodes when each of the two words has many different senses. To prevent the graph size from becoming intractable, we harness the already acquired knowledge about the domain and range of each r and consider only those sense pairs that fall into the previously computed domain and range population, respectively. This yields an enormous pruning effect, and makes the difference between a hopelessly intractable graph and a practically viable method.

Graph Construction. We construct a graph for subsequent Label Propagation analogously to the method for range population (Section 3). There are two differences from the previous graphs. First, every node is a word pair instead of word. Second, there is an additional candidate node pruning step based on domain

and range as described earlier. We capture these by constructing graphs of the following form for each relation.

Definition [Assertion Graph (AG)]:
The AG of a relation r is a weighted undirected graph with nodes V_{AG} and edges E_{AG} as follows:

- V_{AG} consists of all word-level assertion candidates and all sense-level pairs that are not pruned by testing against the domain and range of r (see above).
- E_{AG} consists of three kinds of edges:
  - edges between two word-level assertions,
  - edges between a word-level assertion and a sense-level assertion, and
  - edges between two sense-level assertions.

For an example, see the right side of Figure 1.

Edge Weighting. For all three types of edges, we compute edge weights between two assertions $\langle n_1, r_1, a_1\rangle$ and $\langle n_2, r_2, a_2\rangle$ by considering the similarity between n_1 and n_2 and the similarity between a_1 and a_2. Here, n_1 and n_2 may be either nouns or noun senses, and similarly a_1 and a_2 may be either adjectives or adjective senses. In all cases, we use the multiplicative score

$$\mathrm{sim}(n_1, n_2) \cdot \mathrm{sim}(a_1, a_2)$$

as the edge weight. For example, (`car-n`^{1}, `red-a`^{1}) is similar to tuples like (`car-n`^{1}, `pink-a`^{1}), (`vehicle-n`^{1},`red-a`^{1}), (`bus-n`^{1}, `colorful-a`^{1}).

The individual noun-noun, noun-noun sense, and noun sense-noun sense similarities (all denoted by $\mathrm{sim}(n_1, n_2)$ here) are computed just as for the different types of edge weights earlier in Section 4. Similarly, the adjective-adjective, adjective-adjective sense, and adjective sense-adjective sense similarities (all denoted by $\mathrm{sim}(a_1, a_2)$) are computed just as for the Range Population Graph's edge weights, described earlier in Section 3.

Since there are $O(|E_{\mathrm{AG}}|^2)$ possible assertion edges, we use top-k retrieval methods to efficiently aggregate scores from multiple ranked lists and avoid computing similarities below a threshold.

Label Propagation (LP). For seeds, we consider all assertions where both the noun and the adjective are unambiguous, either because they have only one sense each in WordNet or because our domain- and range-based pruning left us with only one sense pair for the two words. Again, 30 % of the remaining seeds are used for tuning the parameters of the MAD algorithm. Based on these seeds, MAD computes scores for candidate assertions. We accept all assertions for r whose score exceed the dummy label score.

6. EXPERIMENTS

Table 1 summarizes the size of the WebChild knowledge base: the number of distinct senses and assertions, the number of instances of noun and adjectives senses (where a noun or adjective sense that occurs in k different relations counts k times), and the precision estimated by extensive sampling (see below). Table 2 illustrates WebChild by anecdotal examples for range, domain, and assertions. These are top-ranked results, based on a simple scoring function that rewards many occurrences as well as occurrences with multiple distinct patterns.

	#distinct	#instances	Precision
Noun senses	78,077	221,450	0.80
Adj. senses	5,588	7,783	0.90
Assertions	4,649,471	4,649,471	0.82

Table 1: WebChild statistics

We conducted extensive experiments to assess the viability of our approach and the quality of the resulting commonsense relations. Our experiments cover the three tasks addressed in this paper: Subsection 6.1 reports on the quality of relation ranges, Subsection 6.2 presents results on relation domains, and Subsection 6.3 discusses the quality of sense-disambiguated assertions. For each task, we compare against various baseline competitors.

6.1 Relation Ranges

Baselines. Since there is no direct competitor (see related work in Section 8), we designed several baselines as follows.

WordNet attributes: For some relations, WordNet provides the range directly by its `attribute` relation (e.g., `size` contains the adjective senses `small-a`^{1} and `big-a`^{1}).

WordNet attributes expanded: We expanded the above data by including related word senses using synonyms, antonyms, `similar_to`, and `derivationally_related` links. We then iterated this step once more to enlarge the set, but stopped at this point to curb the inevitable topic drift.

WordNet glosses: WordNet provides a short gloss for each adjective. If the gloss mentions a relation, we include the adjective sense in the relation's range. For example, the gloss of `red-a`^{1} mentions the word *color*.

Controlled LDA MFS: Hartung et al. [14] developed a method for creating pseudo-documents per relation r (e.g. color) using nouns and adjectives that appear in the relation. An LDA model estimates the probability $P[a|d]$ for an adjective a given a pseudo-document d, thereby approximating $P[a|r]$ to $P[a|d]$. All adjectives above a threshold for this probability form the range of the relation. We map these adjectives to their most frequent sense (MFS) according to WordNet.

Google Sets MFS: This service, now part of the spreadsheet processor of `docs.google.com`, expands sets of similar words given a few seeds to start with. We use it to find, for each relation, large candidate sets, using five WordNet adjectives as seeds. The resulting adjectives are mapped to the most frequent sense according to WordNet.

Results. We constructed a random sample of 30 adjectives for each relation from the output of WebChild (a total of 570 samples). These were manually evaluated by three people. The kappa value for inter-annotator agreement was 0.869. We likewise drew samples from the outputs of the baseline competitors (or used the entire output when less than 30), and manually assessed them, too. For statistical significance, we computed Wilson score intervals for $\alpha = 95\%$ [8].

The results of this evaluation are shown in Table 3, reporting the macro-averaged precision and the total number of results (coverage). WebChild stands out in this comparison: It discovers far more (sense-mapped) adjectives than any other method, and achieves a very good precision of 90%. WebChild's coverage is three times larger than that of the best prior method [14].

relation	range	domain	assertions
hasTaste	sweet-a[1]	strawberry-n[1]	(chocolate-n[1],creamy-a[2])
	hot-a[9]	chili-n[1]	(pizza-n[1],delectable-a[1])
	sour-a[2]	salsa-n[1]	(salsa-n[1],spicy-a[2])
	salty-a[3]	sushi-n[1]	(burger-n[1],tasty-a[1])
	lemony-a[1]	java-n[2]	(biscuit-n[2],sweet-a[1])
hasShape	triangular-a[1]	leaf-n[1]	(palace-n[1],domed-a[1])
	meandering-a[1]	circle-n[1]	(table-n[2],flat-a[1])
	crescent-a[1]	ring-n[8]	(jeans-n[2],tapered-a[1])
	obtuse-a[2]	egg-n[1]	(tv-n[2],flat-a[1])
	tapered-a[1]	face-n[1]	(lens-n[1],spherical-a[2])

Table 2: Anecdotal example results for `hasTaste`, and `hasShape`

Method	Precision	Coverage
WordNet attributes	1.00	40
WordNet attributes expanded	0.61 ± 0.03	5,145
WordNet glosses	0.70 ± 0.06	3,698
Controlled LDA [14] MFS	0.30 ± 0.06	2,775
Google Sets MFS	0.27 ± 0.04	426
WebChild	0.90 ± 0.03	7,783

Table 3: Results for range population

6.2 Relation Domains

Baselines. We compare WebChild against the following competitors.

Extraction Unambiguous: [2] and [13] manually defined eight patterns (e.g. "the <adj> of <noun> was") to populate the domain of a relation. We applied these patterns to the N-gram corpus and to WordNet glosses. As this technique yields nouns rather than noun senses, we consider only unambiguous nouns with a single sense in WordNet.

Extraction MFS: For higher coverage, we considered all nouns obtained by the previous method and mapped them to their most frequent sense according to WordNet.

Controlled LDA MFS: Using the method of [14] (see baselines on range population) we collect nouns n with a probability $P[n|d]$ above a threshold. We map the nouns to their most frequent senses in WordNet.

WordNet glosses: If a relation name (e.g. *color*) appears in the WordNet gloss of a noun sense, we capture the noun sense as an instance of the relation's domain.

WebChild adj. projections: Our candidate gathering step extracted a large set of noun-adjective pairs from the Web. Since WebChild already has mapped adjectives to specific relations' ranges, a heuristic technique is to assign the co-occurring nouns to the domains of the same relations. Since these nouns are not yet disambiguated, we map them to the most frequent sense in WordNet.

Google Sets: For domain population, this technique performed very poorly due to heterogeneity of seeds; so we do not show any results below.

Results. Table 4 shows the results of this comparison. Again, WebChild stands out, especially by its high coverage. At the same time, its precision of 83% is still fairly high. The method based on WordNet glosses performed slightly better in terms of precision, but yields an order of magnitude lower coverage.

Method	Precision	Coverage
Extraction Unambiguous	0.76 ± 0.06	6,190
Extraction MFS	0.75 ± 0.05	30,445
Controlled LDA [14] MFS	0.71 ± 0.06	9,632
WordNet glosses	0.86 ± 0.03	14,328
WebChild adj. projections	0.71 ± 0.03	175,480
WebChild	0.83 ± 0.03	221,450

Table 4: Results for domain population

6.3 Assertions

As for the main task on commonsense knowledge acquisition, computing fine-grained assertions between noun senses and adjective senses, we can compare WebChild's performance directly with the prior method Controlled LDA (C-LDA) of [14]. C-LDA treats the task as a classification problem, with relations as the classifier's labels. We use the same data that the experiments of [14] were based on. As C-LDA works at the word rather than word-sense level, for the results of their system, a noun-adjective pair is counted as correct if there is at least one sense pair for which the relation label is meaningful. In contrast, we give WebChild the significant disadvantage of considering an assertion as correct only if the senses are correctly chosen, too. Table 5 shows the results of this experiment, demonstrating the clear superiority of WebChild over the prior state of the art.

Baselines. For more comprehensive studies, on our Web-scale candidates, we again designed a variety of baseline competitors.

Controlled LDA MFS: We use C-LDA [14] to map a `hasProperty` candidate pair onto fine-grained relations. Nouns and adjectives are mapped to their most frequent senses in WordNet.

Vector Space MFS: This is analogous to the previous baseline, except that we use the vector space model of [13] rather than LDA.

	Precision	Recall
Controlled LDA [14]	0.33	0.23
WebChild	0.93	0.50

Table 5: Results for assertions on data of [14]

Web Unambiguous Adjective: We consider only those noun-adjective pairs where the adjective has a single sense. We use WebChild's range knowledge to map the adjective to one or more relations. The noun is mapped to its most frequent sense in WordNet.

WebChild Independence: For a given relation, we consider all combinations of noun senses from the domain and adjective senses from the range as assertions for the relation.

Results. Table 6 shows the results of the comparison. WebChild yields more than 4 million assertions at a good precision of 80%. It outperforms all competitors by a large margin, with ten times higher coverage and twice better precision than the best of the prior methods [13]. Interestingly, even the relatively crude WebChild Independence technique performs better than the other baselines. However, its precision is far behind that of the full WebChild method.

Method	Precision	Coverage
Controlled LDA MFS	0.35 ± 0.06	254,576
Vector Space MFS	0.40 ± 0.09	355,018
Web Unambiguous Adjective	0.54 ± 0.09	709,337
WebChild Independence	0.62 ± 0.06	3,399,312
WebChild	0.82 ± 0.03	4,709,149

Table 6: Results for assertions

Table 7 shows the results of WebChild, per relation.

relation	precision	coverage
ability	0.80 ± 0.10	90,288
appearance	0.95 ± 0.05	365,201
beauty	0.70 ± 0.15	95,838
color	0.70 ± 0.15	494,380
emotion	0.90 ± 0.09	79,630
feeling	0.91 ± 0.08	141,453
length	0.70 ± 0.15	90,021
motion	0.80 ± 0.10	146,148
smell	0.82 ± 0.10	25,347
quality	0.82 ± 0.10	793,484
sensitivity	0.70 ± 0.15	5,727
shape	0.80 ± 0.10	359,789
size	0.82 ± 0.10	910,901
sound	0.71 ± 0.15	130,952
state	0.88 ± 0.09	563,022
strength	0.82 ± 0.10	165,412
taste	0.70 ± 0.15	19,892
temperature	0.80 ± 0.13	27,399
weight	0.70 ± 0.15	144,587
overall	0.82 ± 0.03	4,709,149

Table 7: Quality of WebChild relations

7. USE-CASE: POPULATING CLASSES

As a use-case that demonstrates the application benefits of WebChild, we studied the problem of populating semantic classes, such as `river`, `car`, or `singer`. This problem is often addressed as a set-expansion task [36]: Given a small number of seeds, which are instances (or hyponyms) of the same class, find as many additional instances as possible and rank them into a high-precision list. For example, for seeds like *Mississippi*, *Nile*, and *Ganges*, we would like to collect other rivers such as *Danube*, *Rhine*, *Seine*, *Mekong*, etc. A good baseline to compare with is the *Google Sets* tool, which is part of the `docs.google.com` service. Other methods like [9] may be better, but they are also much more complex and need extensive Web data not available to us.

Our method for class population is fairly simple; its main point is the way it harnesses the WebChild knowledge. For a given noun sense `n` corresponding to an instantiable semantic class, we perform the following steps:

1. We select the m highest ranked adjective senses $\texttt{a}_1, \ldots, \texttt{a}_m$ that are connected with `n` by one or more of WebChild's fine-grained assertions. As these are senses, we can further expand them by their WordNet synonyms, thus constructing a ranked list of adjectives $a_1, \ldots, a_l$ (where usually $l \geq m$), now cast into simple words.
2. To avoid extensively used adjectives of unspecific or ambiguous nature (e.g., *great*), we compute PMI scores between the starting noun *n* and each of the adjectives a_i ($i = 1 \ldots l$):
$$\text{PMI}(n, a_i) = \log_2 \frac{P[n \wedge a_i]}{P[n]\ P[a_i]}$$
We prune all adjectives from the ranked list whose PMI score is below a specified threshold. From the remaining adjectives, we take the top k words $a_1, \ldots, a_k$ (e.g., with $k = 10$).
3. Now we apply the linking-verb patterns that we introduced in Section 2 to the Google N-gram corpus and extract noun phrases from the matching N-grams. This yields frequencies for (n,a_i) co-occurrences. The noun phrases are the candidates for populating the class denoted by *n*.
4. We rank the collected noun phrases *p* by aggregating over the co-occurrence frequencies:
$$\text{score}(p) = \sum_{i=1}^{k} \text{freq}(p, a_i) \times \text{weight}(a_i)$$
where $\text{weight}(a_i)$ is the score of the original $\texttt{a}_i$ for noun sense `n` in WebChild (based on pattern-matching statistics, see Section 3).

As a demonstration of the high quality that our method achieves, we evaluate its *precision@5*, in comparison to the top-5 results from Google Sets. We did this for the following 10 test cases (5 common nouns and 5 proper nouns): `river`, `ice cream`, `mountain`, `chocolate`, `keyboard`, `nile river`, `bangalore`, `tiger lily`, `parsley`, `florine`. We evaluate Google Sets with 1 seed (G-1) and 2 seeds (G-2) against WebChild, which only takes the class noun as input (W-1). G-1 runs into limitations, but G-2 performs reasonably well even in this extreme situation. For example, with seed *river* as input, G-1 gives as output *boca*, *estudiantes*, *independiente*, *racing*, *san lorenzo*; with the seed *tiger lily* as input, G-1 produces no output. G-2, with the seeds *river*, *river valley*, gives as output *canyon*, *arizona*, *valley*, *colorado*; with the seeds *tiger lily*, *panther lily* as input, G-2 gives as output *peacock iris*, *meadow saffron*, *pancratium*, *peruvian lily*, *flag*.

Table 8 shows the results. WebChild outperforms G-1 and G-2 on common nouns. On proper nouns, G-2 outperforms WebChild, but WebChild performs as well as G-1. Tables 9 and 10 show the top-10 WebChild adjectives, and the top-5 set expansions for the input `chocolate` and `keyboard` respectively.

Approach	Genre	P@5
G-1	common noun	0.52
G-2	common noun	0.72
W-1	common noun	**0.92**
G-1	proper noun	0.52
G-2	proper noun	**0.68**
W-1	proper noun	0.52

Table 8: Results for set expansion

top-10 adjectives	`smooth`, `assorted`, `dark`, `fine`, `delectable`, `black`, `decadent`, `white`, `yummy`, `creamy`
top-5 expansions	*chocolate bar*, *chocolate cake*, *milk chocolate*, *chocolate chip*, *chocolate fudge*

Table 9: `chocolate`: top-10 adj, top-5 expansions

top-10 adjectives	`ergonomic`, `foldable`, `sensitive`, `black`, `comfortable`, `compact`, `lightweight`, `comfy`, `pro`, `waterproof`
top-5 expansions	*keyboard*, *usb keyboard*, *computer keyboard*, *qwerty keyboard*, *optical mouse*, *touch screen*

Table 10: `keyboard`: top-10 adj, top-5 expansions

8. RELATED WORK

Large-scale knowledge bases have received much attention in recent years, most notable endeavors being Freebase [6], DBpedia [3], and Yago [33]. These and other projects along similar lines focus on facutal knowledge, and disregard the kind of commonsense knowledge considered in this paper. Methods for open information extraction have harnessed verbal phrases and patterns derived from them [27], [10], [26]; however, their focus is still on factual knowledge and does not deliver a hierarchy of disambiguated commonsense properties.

The seminal Cyc project [21] aimed to manually compile commonsense knowledge, with emphasis on rules rather than properties. The publicly available version OpenCyc does not provide any properties of the kind addressed here. The WordNet thesaurus [12] contains a small amount of properties; we use these as seeds for WebChild and we also compare the WebChild results against using WordNet alone.

ConceptNet [23, 15, 31] is probably the largest repository of commonsense assertions about the world, covering relations such as `hasProperty`, `usedFor`, `madeOf`, `motivatedByGoal`, etc. Ongoing projects on extending ConceptNet have made use of crowdsourcing [15, 30] and pattern-based extraction from Web pages [35]. [1] and [16] present games-with-a-purpose to acquire commonsense facts. Approaches for extracting noun properties from text include pattern-based information extraction [2] and corpus co-occurrence analysis [4]. However, this line of research stayed at the level of a single generic `hasProperty` relation.

There is little prior work on methods for acquiring fine-grained commonsense relations. [2] proposed patterns like (*<object> is a/the <attribute> of <subject>*, e.g. *brown is a color of dogs*) to find more specific properties, but even on the Web this method yields only very low recall [4]. [13, 14] developed classifiers for mapping assertion candidates into a small set of 8 fine-grained relations, using vector-space features as well as LDA-based topic models. However, these methods assume that the given assertions are already correct instances of at least the generic `hasProperty` relation. So this work tackled the assertion classification problem, not the problem of computing assertions from raw data. [20] studied relations like `hasSize`, `hasShape`, etc., assigning their instances to word senses. However, it solely relied on human elicitation for 50 nouns, and is not suited for automation at large scale. Finally, [5] used WordNet and ConceptNet to infer commonsense rules, e.g., edible objects are likely to be found at a supermarket. This task is quite different from acquiring commonsense properties.

Unlike most of the above works, our method yields sense-disambiguated assertions for fine-grained relations. In contrast to all prior work, our method is able to produce large-scale output of high quality.

9. CONCLUSION

We presented WebChild, the first comprehensive commonsense knowledge base with fine-grained relations about sense-disambiguated nouns and adjectives. Our methodology combines pattern-based candidate gathering from Web corpora with semi-supervised Label Propagation over judiciously constructed weighted graphs. Experiments demonstrate that this methodology can achieve high precision with good coverage. WebChild is publicly available as a resource for other researchers.

As for ongoing and future work, we are considering adding value to WebChild by inferring further relationships among word senses: antonyms in addition to those known already in WordNet, and ordinal relations such as `isLargerThan` or `isDarkerThan` (cf. [11]). When applied to image annotations, this could possibly contribute to more semantic training data for computer vision. We are also exploring applications of WebChild. We believe that the rich knowledge of noun-adjective relations is a valuable asset for sentiment mining [24] in general and for analyzing feelings about images [28] in particular.

10. REFERENCES

[1] L. von Ahn, M. Kedia, M. Blum: Verbosity: a Game for Collecting Common-Sense Facts. CHI 2006.

[2] A. Almuhareb, M. Poesio: Attribute-Based and Value-Based Clustering: An Evaluation. EMNLP 2004.

[3] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, Z. Ives: DBpedia: A Nucleus for a Web of Open Data. ISWC/ASWC 2007.

[4] M. Baroni, R. Zamparelli: Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space. EMNLP 2010.

[5] E. Blanco, H. C. Cankaya, and D. Moldovan: Commonsense Knowledge Extraction using Concepts Properties. FLAIRS 2011.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. SIGMOD 2008

[7] T. Brants, A. Franz: Web 1T 5-gram Version 1. Linguistic Data Consortium, 2006.

[8] L.D. Brown, T.T. Cai, A. Dasgupta: Interval Estimation for a Binomial Proportion. Statistical Science 16: 101–133, 2001.

[9] B. B. Dalvi, W. W. Cohen, J. Callan. Websets: Extracting Sets of Entities from the Web using Unsupervised Information Extraction. WSDM 2012.

[10] L. Del Corro, R. Gemulla. ClausIE: clause-based open information extraction. WWW 2013.

[11] G. de Melo, M. Bansal: Good, Great, Excellent: Global Inference of Semantic Intensities. Transactions of the ACL, 2013.

[12] C.D. Fellbaum, G.A. Miller (Eds.): WordNet: An Electronic Lexical Database. MIT Press, 1998.

[13] M. Hartung, A. Frank: A Structured Vector Space Model for Hidden Attribute Meaning in Adjective-Noun Phrases. COLING 2010.

[14] M. Hartung, A. Frank: Exploring Supervised LDA Models for Assigning Attributes to Adjective-Noun Phrases. EMNLP 2011.

[15] C. Havasi, R. Speer, J. Alonso: ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge. RANLP 2007.

[16] A. Herdagdelen, M. Baroni: Bootstrapping a Game with a Purpose for Commonsense Collection. ACM TIST 3(4): 59 (2012)

[17] G. Hirst, D. St-Onge: Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In [12], 1998.

[18] IBM Journal of Research and Development 56(3/4), Special Issue on “This is Watson”, 2012.

[19] C. Leacock and M. Chodorow: Combining local context and WordNet similarity for word sense identification. Fellbaum 1998, pp. 265âĂŞ28

[20] G.E. Lebani, E. Pianta: Encoding Commonsense Lexical Knowledge into WordNet. Global WordNet Conference 2012.

[21] Douglas B. Lenat: CYC: A Large-Scale Investment in Knowledge Infrastructure. Comm. of the ACM 38(11), pp. 32-38, 1995.

[22] M. Lesk: Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. SIGDOC 1986.

[23] H. Liu, P. Singh: ConceptNet: a Practical Commonsense Reasoning Toolkit. BT Technology Journal, 2004.

[24] B. Liu: Sentiment Analysis and Opinion Mining. Morgan & Claypool, 2012.

[25] S. Banerjee, T. Pedersen: An Adapted Lesk Algorithm for Word Sense Disambiguation using WordNet. CICLing 2002.

[26] W. Derry, P. Pratim Talukdar, T. Mitchell. PIDGIN: ontology alignment using web text as interlingua. CIKM 2013.

[27] A. Fader, S. Soderland, O. Etzioni. Identifying relations for open information extraction. EMNLP 2011.

[28] S. Siersdorfer, J. Hare: Analyzing and Predicting Sentiment of Images on the Social Web. ACM Multimedia 2010.

[29] A. Singhal: Introducing the Knowledge Graph: Things, Not Strings. `googleblog.blogspot.co.uk`, 16 May 2012.

[30] R. Speer, C. Havasi, H. Surana: Using Verbosity: Common Sense Data from Games with a Purpose. FLAIRS 2010.

[31] R. Speer, C. Havasi: Representing General Relational Knowledge in ConceptNet 5. LREC 2012.

[32] S. Staab, R. Studer (Eds.): Handbook on Ontologies, Springer, 2009.

[33] F.M. Suchanek, G. Kasneci, G. Weikum: YAGO: A Core of Semantic Knowledge. WWW 2007.

[34] P.P. Talukdar, K. Crammer: New Regularized Algorithms for Transductive Learning. ECML/PKDD 2009.

[35] N. Tandon, G. de Melo, G. Weikum: Deriving a Web-Scale Common Sense Fact Database. AAAI 2011.

[36] R.C. Wang, W.W. Cohen: Language-Independent Set Expansion of Named Entities Using the Web. ICDM 2007.

[37] X. Zhu, Z. Ghahramani, J.D. Lafferty: Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. ICML 2003.