Natural Language Semantics using Probabilistic Logic

Islam Beltagy
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
beltagy@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professors: Raymond J. Mooney, Katrin Erk

Abstract

With better natural language semantic representations, computers can do more applications more efficiently as a result of better understanding of natural text. However, no single semantic representation at this time fulfills all requirements needed for a satisfactory representation. Logic-based representations like first-order logic capture many of the linguistic phenomena using logical constructs, and they come with standardized inference mechanisms, but standard first-order logic fails to capture the "graded" aspect of meaning in languages. Distributional models use contextual similarity to predict the "graded" semantic similarity of words and phrases but they do not adequately capture logical structure. In addition, there are a few recent attempts to combine both representations either on the logic side (still, not a graded representation), or in the distribution side(not full logic).

We propose using probabilistic logic to represent natural language semantics combining the expressivity and the automated inference of logic, and the gradedness of distributional representations. We evaluate this semantic representation on two tasks, Recognizing Textual Entailment (RTE) and Semantic Textual Similarity (STS). Doing RTE and STS better is an indication of a better semantic understanding.

Our system has three main components, 1. Parsing and Task Representation, 2. Knowledge Base Construction, and 3. Inference. The input natural sentences of the RTE/STS task are mapped to logical form using Boxer which is a rule based system built on top of a CCG parser, then they are used to formulate the RTE/STS problem in probabilistic logic. Then, a knowledge base is represented as weighted inference rules collected from different sources like WordNet and on-the-fly lexical rules from distributional semantics. An advantage of using probabilistic logic is that more rules can be added from more resources easily by mapping them to logical rules and weighting them appropriately. The last component is the inference, where we solve the probabilistic logic inference problem using an appropriate probabilistic logic tool like Markov Logic Network (MLN), or Probabilistic Soft Logic (PSL). We show how to solve the inference problems in MLNs efficiently for RTE using a modified closed-world assumption and a new inference algorithm, and how to adapt MLNs and PSL for STS by relaxing conjunctions. Experiments show that our semantic representation can handle RTE and STS reasonably well.

For the future work, our short-term goals are 1. better RTE task representation and finite domain handling, 2. adding more inference rules, precompiled and on-the-fly, 3. generalizing the modified closed–world assumption, 4. enhancing our inference algorithm for MLNs, and 5. adding a weight learning step to better adapt the weights. On the longer-term, we would like to apply our semantic representation to the question answering task, support generalized quantifiers, contextualize WordNet rules we use, apply our semantic representation to languages other than English, and implement a probabilistic logic Inference Inspector that can visualize the proof structure.

Contents

1 Introduction

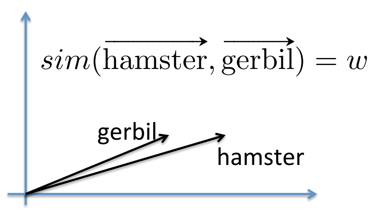
Natural Language semantics is the study of representing the "meaning" of natural text in a machine friendly representation that supports automated reasoning, and that can be acquired automatically from the natural text. Efficient semantic representations (meaning representations) and reasoning tools give computers the power to perform useful complex applications like Question Answering, Automatic Grading and Machine Translation. However, applications and tasks in natural language semantics are very diverse and pose different requirements on the underlying formalism for representing meaning. Some tasks require a detailed representation of the structure of complex sentences. Some tasks require the ability to recognize near-paraphrases or degrees of similarity between sentences. Some tasks require logical inference, either exact or approximate. Often it is necessary to handle ambiguity and vagueness in meaning. Finally, we frequently want to be able to learn relevant knowledge automatically from corpus data.

There is no single representation for natural language meaning at this time that fulfills all requirements, but there are representations that meet some of the criteria. Logic-based representations (Montague, 1970; Kamp & Reyle, 1993) like first-order logic provide an expressive and flexible formalism to deeply express semantics by representing many of the linguistic constructs like conjunctions, disjunctions, negations and quantifiers, and in addition, they come with standardized inference mechanisms. On the other hand, first-order logic fails to capture the "graded" aspect of meaning in languages because it is binary by nature. Distributional models (Turney & Pantel, 2010) use contextual similarity to predict the "graded" semantic similarity of words and phrases (Landauer & Dumais, 1997; Mitchell & Lapata, 2010), and to model polysemy (Schutze, 1998; Erk & Padó, 2008; Thater, Fürstenau, & Pinkal, 2010), but they do not adequately capture logical structure (Grefenstette, 2013). This suggests that distributional models and logic-based representations of natural language meaning are complementary in their strengths (Grefenstette & Sadrzadeh, 2011; Garrette, Erk, & Mooney, 2011), which encourages developing new techniques to combine them. There are a few recent attempts to combine logical and distributional representations. Lewis and Steedman (2013) use distributional information to determine word senses, but still produce a strictly logical semantic representation that does not address the "graded" nature of linguistic meaning. Also Grefenstette (2013) tries to represent all logical constructs using vectors and tensors, but concludes that they do not adequately capture logical structure

We propose a semantic representation that relies on probabilistic logic to combine the advantages of logical and distributional semantics, in which logical form is the primary meaning representation and distributional information is encoded in the form of "weighted" logical rules (Beltagy, Chau, Boleda, Garrette, Erk, & Mooney, 2013). Probabilistic logic frameworks like Markov Logic Networks (MLN) (Richardson & Domingos, 2006) and Probabilistic Soft Logic (PSL) (Kimmig, Bach, Broecheler, Huang, & Getoor, 2012) are Statistical Relational Learning (SRL) techniques (Getoor & Taskar, 2007) that combine logical and statistical knowledge in one uniform framework, and provide a mechanism for coherent probabilistic inference. Probabilistic logic frameworks represent the uncertainty in terms of weights on the logical rules as in the example below.

$$\begin{aligned}&\forall x.\, Smoke(x) \Rightarrow Cancer(x) \mid 1.5\\ &\forall x.y\; Friend(x,y) \Rightarrow (Smoke(x) \Leftrightarrow Smoke(y)) \mid 1.1\end{aligned} \tag{1}$$

The example denotes that if someone smokes, there is a chance that he gets cancer, and the smoking behaviour of friends is usually similar. A probabilistic logic program defines a probability distribution over possible worlds, represented as graphical model, which is then used to draw inferences. Inference in MLNs is intractable and usually exact inference is replaced with sampling techniques. On the other hand, PSL uses continuous truth values for the ground atoms and uses continuous relaxations of the logical operators, then


$sim(\overrightarrow{\text{hamster}}, \overrightarrow{\text{gerbil}}) = w$
gerbil
hamster

$$\forall x\big(\text{hamster}(x) \rightarrow \text{gerbil}(x)\big) \mid f(w)$$

Figure 1: Turning distributional similarity into a weighted inference rule

frames the inference problem as a simple linear program.

Before discussing the components of our semantic representation, we first discuss how to evaluate it. For evaluation, we use two standard tasks, Recognizing Textual Entailment (RTE) (Dagan, Roth, Sammons, & Zanzotto, 2013) and Semantic Textual Similarity (STS) (Agirre, Cer, Diab, & Gonzalez-Agirre, 2012). Given two sentences, RTE is the task of finding out if the first entails, contradicts, or is not related to the second, while STS is the task of finding how semantically similar they are on a scale from 1 to 5. Both tasks require deep understanding to the semantics of the sentences to be able to draw correct conclusions, which serves as a benchmark for the semantic representation. In addition, RTE and STS have many applications like Question Answering, Information Retrieval, Automatic Grading and Machine Translation.

Our approach has three main components, 1. Parsing and Task Representation, where input natural sentences are mapped to logic then used to represent the target task as a probabilistic inference problem, 2. Knowledge Base Construction, where the background knowledge is collected from different sources, encoded as first-order logic rules and weighted. 3. Inference, which solves the generated probabilistic logic problem. Inference is usually the bottleneck in SRL frameworks, because inferences in SRL tend to be intractable problems that do not scale for large problem size. One powerful advantage of relying on probabilistic logic as a semantic representation is that the logic allows for a modular system. This means, the most recent advancements in any of the system components, in parsing, in knowledge base resources, and in inference algorithms, can be easily incorporated in the system.

In the Parsing and Task Representation step, we map input sentences to logic using Boxer (Bos, 2008), a wide-coverage semantic analysis tool built on top of a CCG parser (Clark & Curran, 2004). We show how to use the logical formulas to formulate the RTE and STS tasks as probabilistic logic inference problems. RTE performs two inferences because it is a three-way classification task, and STS is treated as two entailments tasks, from the first sentences to the second, and from the second to the first (Beltagy, Erk, & Mooney, 2014a). It is important to note that probabilistic logic frameworks make the Domain Closure Assumption (DCA) which states that there are no objects in the universe other than the named constants (Richardson & Domingos, 2006). This means, constants and entities need to be explicitly introduced in the domain in a way that makes probabilistic logic produce the expected inferences. We introduce new constants and entities in the domain through skolemization and pragmatic analysis of the sentences in order to avoid having an empty domain and to have universal quantifiers behave as expected in standard first-order logic.

In the Knowledge Base Construction step, we collect "on-the-fly" rules generated from distributional semantics, capturing semantic similarities between words (Beltagy et al., 2013), and that is how we encode the distributional information in our semantic representation. Rules are weighted, and the weight is a function of the semantic similarity score between the words. Figure 1 shows an example of such rule. We also add hard rules (infinite weight) from WordNet (Princeton University, 2010) for Synonyms, Hypernyms, and Antonyms, which experiments showed to be a valuable resource.

In the Inference step, first, we show how to perform the RTE task using MLNs and adapt inference to allow it to scale. We implement an MLN inference algorithm that supports querying complex logical formula, which is not supported in the available MLN tools (Beltagy & Mooney, 2014). Then, we enforce a modified closed-world assumption that helps reduce the size of the inference problem and make inference tractable (Beltagy & Mooney, 2014).

Second, we show how to perform the STS task on MLNs. The deterministic conjunction in logic is more restrictive than what the STS task needs. Therefore, we replace the deterministic conjunction with an average combiner (Natarajan, Khot, Lowd, Tadepalli, Kersting, & Shavlik, 2010) that is less strict than the conjunction, and more suitable for the STS task (Beltagy et al., 2013). Third, we show how to perform the STS task using PSL which is shown to be faster than MLNs and more suitable for the STS task. We show how to adapt PSL for the STS task by replacing the conjunction with an averaging function, and a heuristic grounding algorithm (Beltagy et al., 2014a). Finally, we present the evaluation of our system for the RTE and STS tasks (Beltagy, Erk, & Mooney, 2014b), which shows that our semantic representation is able to handle both tasks reasonably well.

In the short-term, we propose to extend our work in the following directions:

- Better RTE task formulation: We detect contradictions with the help of an additional inference, however, this inference misses some contradictions. We need a different inference that can capture contradictions more accurately. Also we propose that we replace each inference $P(Q|E)$ with the ratio $\dfrac{P(Q|E)}{P(Q)}$ which indicates to what extent adding E changes probability of Q, which is more informative than $P(Q|E)$ alone.

- DCA and Negated Existential: Our handling of quantifiers with the DCA is missing handling of negated existential queries. We need to add support for this form of universal quantifiers to get correct inferences.

- Paraphrase Rules: Large collections of paraphrases like PPDB (Ganitkevitch, Van Durme, & Callison-Burch, 2013) are available. We will translate these rules to logic and add them to our knowledge base.

- Distributional phrasal rules: In addition to the lexical distributional rules we have, we will add rules between short phrases. Phrases are defined using a set of predefined templates.

- More efficient MLN inference for complex queries: Currently, our inference algorithm performs inference by estimating the partition function of two different graphical models. It would be more efficient to perform both estimates in one inference step exploiting the similarities between the two graphical models, and that we are only interested in the ratio between the two partition function not their absolute values.

- Generalize the modified closed-world assumption: The one we use in our system so far assumes a predefined form of inference rules. We want to generalize the definition of the modified closed-world to arbitrary forms of rules.

- Weight Learning: Weight learning can be useful in many ways in our system. For example, it can be used to learn better weights on inference rules, and to assign different weights to different parts of the sentence in the STS task. We would like to apply weight learning to at least one of these problems.

In the long-term, we propose to extend our work in the following directions:

- Question Answering: we would like to apply our semantic representation to the question answering task. In question answering, we search for the answer of a WH question in a large corpus of *unstructured* text. It is an interesting challenge to scale probabilistic logic inference to such large problems.
- Generalized Quantifiers: Generalized quantifiers like Few, Most, Many .. etc (Barwise & Cooper, 1981), are not natively supported in first-order logic, so we would like to add support for them in our system by reasoning about the direction of entailment between parts of the pair of RTE sentences. Few and Most can also be represented by replacing then with Every, and set a non-infinite weight for the rule indicating that some worlds violate it.
- Contextualized WordNet rules: We would like to replace WordNet's hard rules with a weighted rules for different senses of the words, where rule's weight comes from Word Sense Disambiguation (WSD) step. This way, we take the context and the ambiguity of the words into account.
- Other Languages: we would like to see how our semantic representation be applied for languages other than English. Theoretically, the proposed semantic representation is language independent, but practically, not all the resources and tools are available, especially CCG parser and Boxer.
- Inference Inspector: this is an additional tool added to the Probabilistic logic inference process. It gives insights on how the inference process goes, and outputs the rules with the biggest impact on the inference's result. In MLNs, all rules have some impact on the result, so finding the most impactful ones is not straight forward. In the RTE task, this inspector can help finding what parts of T entail what parts of H, and what rules are used. This way we can analyse RTE pairs easily to find the missing rules.

2 Background and Related Work

2.1 Logical Semantics

Logic-based representations of meaning have a long tradition in natural language (Montague, 1970; Kamp & Reyle, 1993). They handle many complex semantic phenomena such as relational propositions, logical operators, and quantifiers; however, standard first-order logic and theorem provers are binary in nature which prevents them from capturing the "graded" aspects of meaning in language. Also, it is difficult to construct formal ontologies of properties and relations that have broad coverage, and mapping sentences into logical expressions utilizing such an ontology is very difficult (Bos, 2013). Consequently, current logical semantic analysis systems are mostly restricted to quite limited domains, such as querying a specific database (Kwiatkowski, Choi, Artzi, & Zettlemoyer, 2013; Berant, Chou, Frostig, & Liang, 2013). In contrast, our system is not limited to any formal ontology as we use a wide-coverage tool for semantic analysis.

Boxer (Bos, 2008) is a software package for wide-coverage semantic analysis that produces logical forms using Discourse Representation Structures (Kamp & Reyle, 1993). It builds on the C&C CCG parser (Clark & Curran, 2004). which maps the input sentences into a lexically-based logical form, in which the predicates are words in the sentence. For example, the sentence "A man is driving a car" in logical form is:

$$\exists x, y, z.\, man(x) \land agent(y, x) \land drive(y) \land patient(y, z) \land car(z) \tag{2}$$

2.2 Distributional Semantics

Distributional models (Turney & Pantel, 2010), on the other hand, use statistics on contextual data from large corpora to predict semantic similarity of words and phrases (Landauer & Dumais, 1997; Mitchell & Lapata, 2010). They are motivated by the observation that semantically similar words occur in similar contexts, so words can be represented as vectors in high dimensional spaces generated from the contexts in which they occur (Landauer & Dumais, 1997; Lund & Burgess, 1996). Such models have also been extended to compute vector representations for larger phrases, e.g. by adding the vectors for the individual words (Landauer & Dumais, 1997) or by a component-wise product of word vectors (Mitchell & Lapata, 2008, 2010), or more complex methods that compute phrase vectors from word vectors and tensors (Baroni & Zamparelli, 2010; Grefenstette & Sadrzadeh, 2011). Therefore, distributional models are relatively easier to build than logical representations, automatically acquire knowledge from "big data", and capture the "graded" nature of linguistic meaning, but they do not adequately capture logical structure (Grefenstette, 2013).

2.3 Probabilistic Logic

Probabilistic logic frameworks are Statistical Relational Learning (SRL) techniques (Getoor & Taskar, 2007) that combine logical and statistical knowledge in one uniform framework, and provide a mechanism for coherent probabilistic inference. Probabilistic logic frameworks typically employ weighted formulas in first-order logic to compactly encode complex probabilistic graphical models. Weighting the rules is a way of softening them compared to hard logical constraints and thereby allowing situations in which not all clauses are satisfied. Equation 1 is an example of the weighted logical rules. With the weighted rules, a set of constants need to be specified. For the rules in equation 1, we can add constants representing two persons, Anna (A) and Bob (B). Probabilistic logic uses the constants to "ground" atoms with variables, so we get "ground atoms" like $Smoke(A)$, $Smoke(B)$, $Cancer(A)$, $Cancer(B)$, $Friend(A, A)$, $Friend(A, B)$, $Friend(B, A)$, $Friend(B, B)$. Rules are also grounded by replacing each atom with variables with all its possible ground atoms. A probabilistic logic program defines a probability distribution over the possible values of the ground atoms where they are treated as random variables. In addition to the set of rules R, a probabilistic logic program takes an evidence set E asserting some truth values about some of the random variables, e.g. $Cancer(A)$ means that Anna has cancer. Then, given a query formula Q, probabilistic logic inference calculates the probability $P(Q|R, E)$ which is the answer to the query.

2.3.1 Markov Logic Network

Markov Logic Networks (MLN) (Richardson & Domingos, 2006) are one of the probabilistic logic frameworks. MLNs define a probability distribution over possible worlds, where a world's probability increases exponentially with the total weight of the logical clauses that it satisfies. Probability of a given world x is denoted by:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i (x)\right) \tag{3}$$

where Z is the partition function, i ranges over all formulas F_i is the MLN, w_i is the weight of F_i and $n_i(x)$ is the number of true groundings of F_i in the world x. MLN's marginal inference calculates the probability $P(Q|E, R)$, where Q is a query, E is the evidence set, and R is the set of weighted formulas.

Alchemy (Kok, Singla, Richardson, & Domingos, 2005) is the most widely used MLN implementation. It is a software package that contains implementations of a variety of MLN inference and learning algorithms. However, developing a scalable, general-purpose, accurate inference method for complex MLNs is an open problem.

2.3.2 Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) is a recently proposed alternative framework for probabilistic logic (Kimmig et al., 2012; Bach, Huang, London, & Getoor, 2013). It uses logical representations to compactly define large graphical models with "continuous" variables, and includes methods for performing efficient probabilistic inference for the resulting models. A key distinguishing feature of PSL is that ground atoms have soft, continuous truth values in the interval [0, 1] rather than binary truth values as used in MLNs and most other probabilistic logics. Given a set of weighted logical formulas, PSL builds a graphical model defining a probability distribution over the continuous space of values of the random variables in the model. A PSL model is defined using a set of weighted if-then rules in first-order logic, as in the following example:

$$\begin{aligned} &\forall x, y, z.\, friend(x, y) \wedge votesFor(y, z) \Rightarrow votesFor(x, z) \mid 0.3 \\ &\forall x, y, z.\, spouse(x, y) \wedge votesFor(y, z) \Rightarrow votesFor(x, z) \mid 0.8 \end{aligned} \tag{4}$$

The first rule states that a person is likely to vote for the same person as his/her friend. The second rule encodes the same regularity for a person's spouse. The weights encode the knowledge that a spouse's influence is greater than a friend's in this regard.

In addition, PSL includes *similarity functions*. Similarity functions take two strings or two sets as input and return a truth value in the interval [0, 1] denoting the similarity of the inputs. For example, this is a rule that incorporate the similarity of two predicates:

$$\forall x.\, similarity(\text{“}predicate1\text{”}, \text{“}predicate2\text{”}) \wedge predicate1(x) \Rightarrow predicate2(x) \tag{5}$$

As mentioned above, each ground atom, a, has a soft truth value in the interval [0, 1], which is denoted by $I(a)$. To compute soft truth values for logical formulas, Lukasiewicz's relaxation of conjunctions($\wedge$), disjunctions($\vee$) and negations($\neg$) are used:

$$\begin{aligned} I(l_1 \wedge l_1) &= max\{0, I(l_1) + I(l_2) - 1\} \\ I(l_1 \vee l_1) &= min\{I(l_1) + I(l_2), 1\} \\ I(\neg l_1) &= 1 - I(l_1) \end{aligned} \tag{6}$$

Then, a given rule $r \equiv r_{body} \Rightarrow r_{head}$, is said to be *satisfied* (i.e. $I(r) = 1$) iff $I(r_{body}) \leq I(r_{head})$. Otherwise, PSL defines a *distance to satisfaction* $d(r)$ which captures how far a rule r is from being satisfied: $d(r) = max\{0, I(r_{body}) - I(r_{head})\}$. For example, assume we have the set of evidence:
$I(spouse(B, A)) = 1$, $I(votesFor(A, P)) = 0.9$, $I(votesFor(B, P)) = 0.3$, and that r is the resulting ground instance of rule (4). Then $I(spouse(B, A) \wedge votesFor(A, P)) = max\{0, 1 + 0.9 - 1\} = 0.9$, and $d(r) = max\{0, 0.9 - 0.3\} = 0.6$.

Using *distance to satisfaction*, PSL defines a probability distribution over all possible interpretations I of all ground atoms. The pdf is defined as follows:

$$\begin{aligned} p(I) &= \frac{1}{Z} \exp\,[-\sum_{r \in R} \lambda_r (d(r))^p]; \\ Z &= \int_I \exp\,[-\sum_{r \in R} \lambda_r (d(r))^p] \end{aligned} \tag{7}$$

where Z is the normalization constant, λ_r is the weight of rule r, R is the set of all rules, and $p \in \{1, 2\}$ provides two different loss functions. For our application, we always use $p = 1$

PSL is primarily designed to support MPE inference (Most Probable Explanation). MPE inference is the task of finding the overall interpretation with the maximum probability given a set of evidence. Intuitively, the interpretation with the highest probability is the interpretation with the lowest distance to satisfaction. In other words, it is the interpretation that tries to satisfy all rules as much as possible. Formally, from equation 7, the most probable interpretation, is the one that minimizes $\sum_{r \in R} \lambda_r (d(r))^p$. In case of $p = 1$, and given that all $d(r)$ are linear equations, then minimizing the sum requires solving a linear program, which, compared to inference in other probabilistic logics such as MLNs, can be done relatively efficiently using well-established techniques. In case $p = 2$, MPE inference can be shown to be a second-order cone program (SOCP) (Kimmig et al., 2012).

2.4 Tasks

We evaluate our semantic representation using the RTE and STS tasks.

2.4.1 Recognizing Textual Entailment

Recognizing Textual Entailment (RTE) (Dagan et al., 2013) is the task of determining whether one natural language text, the *premise* T, Entails, Contradicts, or not related (Neutral) to another, the *hypothesis* H. Here are examples from the SICK dataset (Marelli, Menini, Baroni, Bentivogli, Bernardi, & Zamparelli, 2014):

- Entailment

T: A man and a woman are walking together through the woods.
H: A man and a woman are walking through a wooded area.

- Contradiction

T: A man is jumping into an empty pool
H: A man is jumping into a full pool

- Neutral

T: A young girl is dancing
H: A young girl is standing on one leg

2.4.2 Semantic Textual Similarity

Semantic Textual Similarity (STS) is the task of judging the similarity of a pair of sentences on a scale from 0 to 5, and was recently introduced as a SemEval task (Agirre et al., 2012). Gold standard scores are averaged over multiple human annotations and systems are evaluated using the Pearson correlation between a system's output and gold standard scores. Here are some examples:

- "A man is playing a guitar." "A woman is playing the guitar.", score: 2.75
- "A woman is cutting broccoli." "A woman is slicing broccoli.", score: 5.00
- "A car is parking." "A cat is playing.", score: 0.00

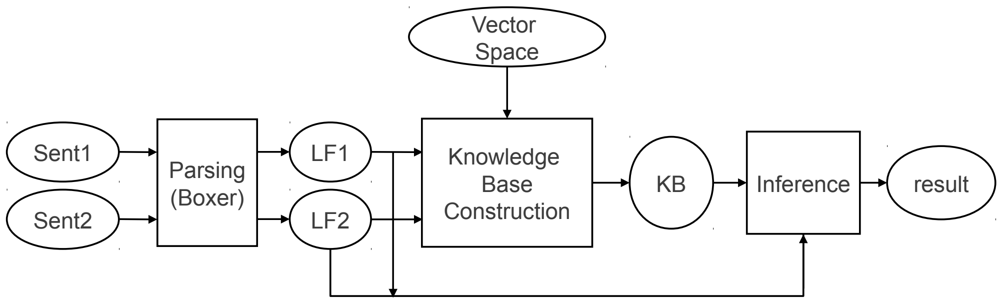


Figure 2: System Architecture

3 Completed Research

This section describes the details of our semantic representation, and how it is used to do the RTE and STS tasks. Figure 2 shows the high level system architecture. Input sentences are mapped to logic using Boxer, the knowledge base KB is collected, then KB and the sentences are passed to the inference engine to solve the inference problem according to the target task.

3.1 Parsing and Task Representation

This is where our system maps natural sentences into logical formulas, then use them to formulate the RTE and STS tasks as probabilistic logic inference problems.

3.1.1 Tasks as Probabilistic Logic Inference

Boxer Natural sentences are mapped to logical form using Boxer (Bos, 2008) as in equation 2. We call Boxer's output alone an uninterpreted logical form because predicates do not have meaning by themselves. They get the meaning from the knowledge base KB we build in section 3.2.

RTE Task We are given two sentences T and H, and we want to find if T entails, contradicts or neutral to H. Checking for entailment in the standard logic is checking if $T \wedge KB \Rightarrow H$, where KB is the knowledge base we build in section 3.2. Its probabilistic version is calculating the probability $P(H|T, KB)$, where H is the probabilistic logic query.

Differentiating between Contradiction and Neutral requires one more inference. It is to calculate the probability $P(H|\neg T, KB)$. In case $Pr(H|T, KB)$ is high, while $Pr(H|\neg T, KB)$ is low, this indicates Entails. In case it is the other way around, this indicates Contradicts. If both values are close, this means T does not affect the probability of H and indicative of Neutral. Practically, we train an SVM classifier with LibSVM's default parameters (Chang & Lin, 2001) to map the two probabilities to the final decision.

STS Task We are given two sentences S_1, S_2 and we want to find how semantically similar they are. We realize the STS task as the two probabilistic entailments $P(S_1|S_2, KB)$ and $P(S_2|S_1, KB)$. The final similarity score is produced from an Additive Regression (Friedman, 1999) model with WEKA's default parameters (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) trained to map the two degree of entailments to a similarity score.

3.1.2 Working with DCA

A significant difference between standard logic and probabilistic logic comes from the fact that probabilistic logic frameworks usually make the Domain Closure Assumption (DCA) (Richardson & Domingos, 2006) which MLNs and PSL make. DCA states that, there are no objects in the universe other than the named constants. This means, constants need to be explicitly introduced in the probabilistic logic program. Constants are used to ground the predicates, and build the graphical model. For different set of constants, a different graphical model is built. For example, constants like Anna A and Bob B need to be explicitly stated along with the rules in equation 1. Without them, the graphical model will be empty (no random variables). Another problem is that DCA changes the semantics of universal quantifiers to operate only on the finite set of constants in the domain. This means that even more constants need to be added to the domain for the universal quantifier to work as expected. This section discusses how we generate constants and entities in the domain for the inference problem to work properly.

Skolemization The first set of constants are introduced through "Skolemization" of T. Skolemizing T replaces non-embedded existentially quantified variables with skolem constants. For example, skolemizing the logical expression in equation 2 is:

$$man(M) \land agent(D, M) \land drive(D) \land patient(D, C) \land car(C) \tag{8}$$

where M, D, C are constants introduced into the domain. In case of embedded existentially quantified variables, they are replaced with skolem functions, where function parameters are the outer universally quantifier variables. For example, here is how the logical form of "All birds fly" is skolemized:

$$\begin{aligned} T &: \forall x.\, bird(x) \Rightarrow \exists y.\, agent(y, x) \land fly(y) \\ skolemized &: \forall x.\, bird(x) \Rightarrow agent(f(x), x) \land fly(f(x)) \end{aligned} \tag{9}$$

The skolem functions should map its arguments to new constants. For the example above, the skolem function should introduce a new "flying event" for each "bird" in the domain. We simulate this behaviour by replacing the skolem function with a new predicate and universally quantified variables, then add the extra constants as evidence. The example above would look like:

$$\forall x.\, bird(x) \Rightarrow \forall y.\, skolem(x, y) \Rightarrow agent(y, x) \land fly(y) \tag{10}$$

For now, let's say we have evidence of a "bird" B_1 (we explain how this entity gets introduced later this section). For all possible values of the universally quantified variables (in this example, the variable x, and its possible values are only the constant B_1), we generate evidence for the $skolem$ predicate with new constants in place of skolemized existentially quantified variable (in this case, the variable y). For the example, we generate one atom $skolem(B_1, C_1)$ where C_1 is the newly introduced constant and the atom is simulating that the function $f(x)$ maps constant B_1 to the constant C_1.

Existence Constants introduced through skolemization are not enough to represent T in a way that supports the desired inferences. We need to introduce additional constants and entities for universally quantified variables in T in order for the domain to be non-empty. Linguistically, this can be justified by pragmatics: When we hear "All birds with wings fly" we assume that the hearer thinks that there are birds with wings.

In Boxer's output, two linguistic constructs result into sentences with universally quantified variables. The first is sentences with implications (in case the implication is not negated). The implication has a universally quantified restrictor (left-hand side), and existentially quantified body (right-hand side), e.g. "All birds with wings fly" in logic is:

$$T : \forall x, y.\, bird(x) \wedge with(x, y) \wedge wing(y) \Rightarrow \exists z.\, agent(z, x) \wedge fly(z) \tag{11}$$

Pragmatically, this sentence implies that there exist birds with wings. In general, we can infer the existence of the entities on the universally quantified left-hand side of the implication, and thereby generate the extra entities needed. For each non-negated implication that Boxer generates, we add to the evidence atoms representing the left-hand side of the implication which is always universally quantified. For the example, we generate evidence of a "bird with wings": $bird(B) \wedge with(B, W) \wedge wing(W)$.

The second linguistic construct that results into sentences with universally quantified variables is sentences with negated existence, like "No bird flies" which in logic is:

$$\neg\exists x, y.\, bird(x) \wedge agent(y, x) \wedge fly(y) \tag{12}$$

We do not need to generate any additional entities for the universally quantified variables in this sentence, because the sentence is negating the existence of the entities.

Universal Quantifier DCA changes the semantics of universal quantifiers to operate only on the constants in the domain. This makes universal quantifiers in H sometimes behave in an undesirable way. For example, consider the RTE problems: "T_1: There is a black bird", "T_2: All birds are black" and "H: All birds are black", which in logic are

$$\begin{aligned} T_1 &: \exists x.\, bird(x) \wedge black(x) \\ skolemized\, T_1 &: bird(B) \wedge black(B) \\ T_2 &: \forall x.\, bird(x) \Rightarrow black(x) \\ skolemized\, T_2 &: \forall x.\, bird(x) \Rightarrow black(x) \\ H &: \forall x.\, bird(x) \Rightarrow black(x) \end{aligned} \tag{13}$$

Because of DCA, probabilistic logic concludes that T_1 entails H because H is true for all constants *in the domain* (in this example, the single constant B). While we used Skolemization and Existence to handle the issues in the representation of T, this problem affects H. As we do with the universal quantifiers in T, we also introduce entities for the universally quantified left-hand side of implication in H, but for a different rationale. In the example shown, we introduce evidence of a new bird $bird(D)$. The rational here is that the introduction of a new evidence $bird(D)$ prevents the hypothesis from being judged true for the RTE pair T_1, H. However, for T_2, H the new bird $bird(D)$ will be inferred to be black, in which case we can take the hypothesis to be true.

In case of H with universal quantifiers from negated existentially quantified sentences as in equation 12, they can not be ignored as we do in case of Existence with T. H should be encoded in a way that it can not be entailed unless T explicitly entails it, not because of the assumption that things are false by default. We keep this case for future work as explained in section 4.1.

3.2 Knowledge Base Construction

This section discusses how we collect the rules of the knowledge base KB

3.2.1 WordNet

WordNet (Princeton University, 2010) is a lexical database of words grouped into sets of synonyms. In addition to grouping synonyms, it lists semantic relations connecting groups. We represent the information on WordNet as "hard" logical rules and add them to the system's KB. The semantic relations we use are:

- Synonyms: $\forall x.\, man(x) \Leftrightarrow guy(x)$
- Hyponym: $\forall x.\, car(x) \Rightarrow vehicle(x)$
- Antonyms: $\forall x.\, tall(x) \Leftrightarrow \neg short(x)$

One advantage of using logic for semantic representation is that it is a powerful representation that can represent different semantic relations accurately.

3.2.2 Distributional Semantics

As depicted in figure 1, distributional information can be encoded as weighted inference rules. This is how we bring logical and distributional semantics together. We treat distributional similarity between words as degree of entailment, a move that has a long tradition (e.g., (Lin & Pantel, 2001; Raina, Ng, & Manning, 2005; Szpektor & Dagan, 2008)).

As we present in (Beltagy et al., 2013), for all pairs of words(a, b) where $a \in T$ and $b \in H$, generate the weighted inference rule

$$\begin{aligned} &\forall x.\, a(x) \Rightarrow b(x) \mid f(sim) \\ &sim = \cos(\vec{a}, \vec{b}) \end{aligned} \tag{14}$$

sim is the similarity measure between the vector of the word a and the vector of the word b. We use cosine similarity, but more advanced measures can also be used. f is a mapping function that maps the similarity measure to a weight that fits the probabilistic logic used. Each rule is assigned a weight $w = f(sim)$ that approximates the likelihood of the rule holding. For MLNs, because weights are in the exponent, f is denoted by:

$$w = f(sim) = log(\frac{sim}{1 - sim}) \tag{15}$$

For PSL, $f(sim) = sim$ because PSL has a special construct to represent similarities.

Distributional representations for words are derived by counting co-occurrences in the ukWaC, WaCkypedia, BNC and Gigaword corpora (Beltagy, Roller, Boleda, Erk, & Mooney, 2014). We use the 2000 most frequent content words as basis dimensions, and count co-occurrences within a two word context window. The vector space is weighted using Positive Pointwise Mutual Information (Roller, Erk, & Boleda, 2014).

3.3 Probabilistic Logical Inference

The last component is probabilistic logical inference. We showed in section 3.1 how to represent the tasks as probabilistic inference problems on the form $P(Q|E, R)$, where Q is the query formula, E is the evidence set, and R is a set of rules. This section shows how to solve this inference problem for different tasks using different probabilistic logic frameworks.

3.3.1 RTE using MLNs

MLN's inference is usually intractable, and using MLN's implementations "out of the box" do not work for our application. This section discusses an MLN implementation that supports complex queries Q. It also suggests a form of closed-world assumption that has the effect of dramatically decreasing the problem size, hence making inference fast.

3.3.1.1 Query Formula Current implementations of MLNs like Alchemy (Kok et al., 2005) do not allow queries to be complex formulas, they can only calculate probabilities of ground atoms. This section discusses an inference algorithm for arbitrary query formulas (Beltagy & Mooney, 2014).

Standard Work-Around Although current MLN implementations can only calculate probabilities of ground atoms, they can be used to calculate the probability of a complex formula through a simple work-around. The complex query formula Q is added to the MLN using the hard formula:

$$Q \Leftrightarrow result(D) \mid \infty \tag{16}$$

where $result(D)$ is a new ground atom that is not used anywhere else in the MLN. Then, inference is run to calculate the probability of $result(D)$, which is equal to the probability of the formula Q. However, this approach can be very inefficient for some queries. For example, consider the query Q,

$$Q : \exists x, y, z.\, man(x) \wedge agent(y, x) \wedge drive(y) \wedge patient(y, z) \wedge car(z) \tag{17}$$

This form of existentially quantified formulas with a list of conjunctively joined atoms, is very common in the inference problems we are addressing, so it is important to have efficient inference for such queries. However, using this Q in equation 16 results in a very inefficient MLN. The direction $Q \Leftarrow result(D)$ of the double-implication in equation 16 is very inefficient because the existentially quantified formula is replaced with a large disjunction over all possible combinations of constants for variables x, y and z (Gogate & Domingos, 2011). Generating this disjunction, converting it to clausal form, and running inference on the resulting ground network becomes increasingly intractable as the number of variables and constants grow.

New Inference Method Instead, we propose an inference algorithm to directly calculate the probability of complex query formulas. The probability of a formula is the sum of the probabilities of the possible worlds that satisfy it. Gogate and Domingos (2011) show that to calculate the probability of a formula Q given a probabilistic knowledge base K, it is enough to compute the partition function Z of K with and without Q added as a hard formula:

$$P(Q \mid K) = \frac{Z(K \cup \{(Q, \infty)\})}{Z(K)} \tag{18}$$

Therefore, all we need is an appropriate algorithm to estimate the partition function Z of a Markov network. Then, we construct two ground networks, one with the query and one without, and estimate their Zs using that estimator. The ratio between the two Zs is the probability of Q.

We tried to estimate Z using a harmonic-mean estimator on the samples generated by MC-SAT (Poon & Domingos, 2006), a popular and generally effective MLN inference algorithm, but we found that the estimates are highly inaccurate as shown in (Venugopal & Gogate, 2013). So, the partition function estimator we use is SampleSearch (Gogate & Dechter, 2011). SampleSearch is an importance sampling algorithm that has been shown to be an effective sampling algorithm when there is a mix of probabilistic and deterministic (hard) constraints, a fundamental property of the inference problems we address. Importance sampling in general is problematic in the presence of determinism, because many of the generated samples violate the deterministic constraints, and they get rejected. Instead, SampleSearch uses a base sampler to generate samples then uses backtracking search with a SAT solver to modify the generated sample if it violates the deterministic constraints. We use an implementation of SampleSearch that uses a generalized belief propagation algorithm called Iterative Join-Graph Propagation (IJGP) (Dechter, Kask, & Mateescu, 2002) as a base sampler. This version is available online (Gogate, 2014).

For the example Q in equation 17, in order to avoid generating a large disjunction because of the existentially quantified variables, we replace Q with its negation $\neg Q$, so the existential quantifiers are replaced with universals, which are easier to ground and perform inference upon. Finally, we compute the probability of the query $P(Q) = 1 - P(\neg Q)$. Note that replacing Q with $\neg Q$ cannot make inference with the standard work-around faster, because with $\neg Q$, the direction $\neg Q \Rightarrow result(D)$ suffers from the same problem of the existential quantifiers instead of the other direction $\neg Q \Leftarrow result(D)$.

3.3.1.2 Modified Closed-World Assumption This section explains why our inference problems are difficult for MLN, and why standard lifting techniques are not enough to solve it. Next it discusses the relationship between the traditional low prior on predicates, and our modified closed-world assumption. Finally, it defines our modified closed-world assumption and describes how it is implemented (Beltagy & Mooney, 2014).

Problem Description In the inference problems we address, typically formulas are long, especially the query formula. First-order formulas result in an exponential number of ground clauses, where the number of ground clauses of a formula is $O(c^v)$, where c is number of constants in the domain, and v is number of variables in the formula. For any moderately long formula, the number of resulting ground clauses is infeasible to process in any reasonable time using available inference algorithms. Even recent lifting techniques (Singla & Domingos, 2008; Gogate & Domingos, 2011) that try to group similar ground clauses to reduce the total number of nodes in the ground network, are not applicable here. Lifting techniques implicitly assume that c is large compared to v, and the number of ground clauses is large because c is large. In our case, c and v are typically in the same range, and v is large, and this makes lifting algorithms fail to find similarities to lift.

Low prior In the inference problems we address, as in most MLN applications, all atoms are initialized with a low prior. This low prior means that, by default, all groundings of an atom have very low probability, unless they can be inferred from the evidence and knowledge base. However, we found that a large fraction of the ground atoms cannot be inferred, and their probabilities remain very low. This suggests that these ground atoms can be identified and removed in advance with very little impact on the approximate nature of the inference. As the number of such ground atoms is large, this has the potential to dramatically decrease the size of the ground network. Our modified closed-world assumption was created to address this issue.

Definition Closed-world, open-world and our modified closed-world assumptions are different ways of specifying what ground atoms are initialized to True, False or Unknown. True and False ground atoms are used to construct the appropriate network but are not part of the final ground Markov network. Only Unknown ground atoms participate in probabilistic inference. All ground atoms specified as evidence are known (True or False). The difference between the three assumptions is in the non-evidence ground atoms.

With a closed-world assumption, non-evidence ground atoms are all False. In case of the open-world assumption, non-evidence ground atoms are all Unknown and they are all part of the inference task. In case of our modified closed-world assumption, non-evidence ground atoms are False by default, unless they are *reachable* from any of the evidence, or from a ground atom in an input formula.

Reachability A ground atom is said to be *reachable* from the evidence if there is a way to propagate the evidence through the formulas and reach this ground atom. The same applies for ground atoms specified in an input formula. For example, consider the evidence set E, and clauses r_1, r_2:

$$\begin{aligned} E &: \{\ g(C_1), h(C_2)\ \} \\ r_1 &: \forall x, y.\ g(x) \lor h(y) \lor i(x, y) \\ r_2 &: \forall x, y.\ j(x) \lor k(y) \lor i(x, y) \end{aligned}$$

From r_1, variables x, y can be assigned the constants C_1, C_2 respectively because of the evidence $g(C_1)$, $h(C_2)$. Then, this evidence gets propagated to $i(C_1, C_2)$, so the ground atom $i(C_1, C_2)$ is Unknown. From r_2, the variables x, y can be assigned the constants C_1, C_2 respectively because of the Unknown ground atom $i(C_1, C_2)$, and this gets propagated to $j(C_1)$, $k(C_2)$, so ground atoms $j(C_1)$, $k(C_2)$ are also Unknown. All other ground atoms, except the evidence $g(C_1)$ and $h(C_2)$, are False because they are not reachable from any evidence.

Note that the definition of reachability here (mcw-reachable) is different from the definition of reachability in graph theory (graph-reachable). Nodes can be graph-reachable but not mcw-reachable. For the example above, consider the full ground network of E and r_1, which contains 8 nodes, and 4 cliques. It is a connected graph, and all nodes are graph-reachable from each others. However, as explained in the example, $i(C_1, C_2)$ is the only mcw-reachable node.

Algorithm and Implementation Algorithm 1 describes the details of the grounding process with the modified closed-world assumption applied. Lines 1 and 2 initialize the reachable set with the evidence and any ground atom in R. Lines 3-11 repeatedly propagate evidence until there is no change in the reachable set. Line 12 generates False evidence for all unreachable ground atoms. Line 13 generates all ground clauses, then lines from 14-31 substitute values of the known ground atoms in the ground clauses. Alchemy drops all True and False ground clauses, but this does not work when the goal of the inference algorithm is to calculate Z. Lines from 16-30 describe the change. True ground clauses are dropped, but not False ground clauses. If a False ground clause is a grounding of one of Q's clauses, then $Z = 0$ and there is no need to perform inference since there is no way to satisfy Q given E and R. If there is False hard clause, then this MLN is inconsistent. Otherwise, the False ground clause can be dropped. The resulting list of ground clauses GC are then passed to the inference algorithm to estimate Z.

3.3.2 STS using MLNs

We showed in section 3.1 how to represent STS as an inference problem in the form $P(Q|E, R)$. However, inference in STS is different from that in RTE (Beltagy et al., 2013). Here is an example why they are different:

S_1: $\exists x_0, e_1.\, man(x_0) \land agent(e_1, x_0) \land drive(e_1)$

S_2: $\exists x_0, e_1, x_2.\, man(x_0) \land agent(e_1, x_0) \land drive(e_1) \land patient(e_1, x_2) \land car(x_2)$

Algorithm 1 Grounding with modified closed-world assumption

```
Input  $R$: $\{K \cup Q\}$ set of first-order clauses, where $K$ is the set of clauses from the input MLN, and $Q$ is
    the set of clauses from the query.
Input  $E$: set of evidence (list of ground atoms)
Output : a set of ground clauses with the modified closed-world assumption applied
 1: Add all $E$ to the $reachable$ ground atoms
 2: Add all ground atoms in $R$ to $reachable$
 3: repeat
 4:     for all $r \in R$ do
 5:         $p$ = propagate $reachable$ ground atoms between predicates sharing the same variable
 6:         add propagated ground atoms ($p$) to $reachable$
 7:         if $p$ not empty then
 8:             $changed$ = $true$
 9:         end if
10:     end for
11: until not $changed$
12: Generate $False$ evidence for ground atoms $\notin reachable$ and add them to $E$
13: $GC$ = Use MLN's grounding process to ground clauses $R$
14: for all $gc \in GC$ do
15:     $gc$ = $gc$ after substituting values of known ground atoms in $E$
16:     if $gc = True$ then
17:         drop $gc$
18:     else if $gc = False$ then
19:         if $gc$ is a grounding of one of $Q$'s clauses then
20:             Terminate inference with $Z = 0$
21:         else
22:             if $gc$ is hard clause then
23:                 Error inconsistent MLN
24:             else
25:                 drop $gc$
26:             end if
27:         end if
28:     else
29:         keep $gc$ in $GC$
30:     end if
31: end for
32: return $GC$
```

Calculating $P(S_2|S_1)$ in an RTE manner gives the probability of zero, because there is no evidence for a car, and the hypothesis predicates are conjoined using a deterministic AND. For RTE, this makes sense: If one of the hypothesis predicates is False, the probability of entailment should be zero. For the STS task, this should in principle be the same, at least if the omitted facts are vital, but it seems that annotators rated the data points in this task more for overall similarity than for degrees of entailment. So in STS, we want the similarity to be a function of the number of elements in the hypothesis that are inferable. Therefore, we need to replace the deterministic AND with a different way of combining evidence. We chose to use the average evidence combiner for MLNs introduced by (Natarajan et al., 2010). To use the average combiner, the full logical form is divided into smaller clauses (which we call mini-clauses), then the combiner averages their probabilities. In case the formula is a list of conjuncted predicates, a mini-clause is a conjunction of a single-variable predicate with a relation predicate (as in the example below). In case the logical form contains a negated sub-formula, the negated sub-formula is also a mini-clause. The hypothesis above after dividing clauses for the average combiner looks like this:

$$\begin{aligned} man(x_0) \wedge agent(e_1, x_0) &\Rightarrow result(x_0, e_1, x_2) \mid w \\ drive(e_1) \wedge agent(e_1, x_0) &\Rightarrow result(x_0, e_1, x_2) \mid w \\ drive(e_1) \wedge patient(e_1, x_2) &\Rightarrow result(x_0, e_1, x_2) \mid w \\ car(x_2) \wedge patient(e_1, x_2) &\Rightarrow result(x_0, e_1, x_2) \mid w \end{aligned} \tag{19}$$

where $result$ becomes the query predicate. Here, $result$ has all of the variables in the clause as arguments in order to maintain the binding of variables across all of the mini-clauses. The weights w are the following function of n, the number of mini-clauses (4 in the above example):

$$w = \frac{1}{n} \times log(\frac{\alpha}{1-\alpha}) \tag{20}$$

where α is a value close to 1 that is set to maximize performance on the training data. Setting w this way produces a probability of α for the $result()$ in cases that satisfy the antecedents of *all* mini-clauses. For the example above, the antecedents of the first two mini-clauses are satisfied, while the antecedents of the last two are not since the premise provides no evidence for an object of the verb $drive$. The similarity is then computed to be the maximum probability of any grounding of the $result$ predicate, which in this case is around $\frac{\alpha}{2}$.

The average combiner is very memory consuming since the number of arguments of the $result()$ predicate can become large (there is an argument for each individual and event in the sentence). Consequently, the inference algorithm needs to consider a combinatorial number of possible groundings of the $result()$ predicate, making inference very slow. However, one experiment that is worth trying is applying the modified closed-world assumption discussed in 3.3.1.2, which potentially can reduce the number of groundings.

3.3.3 STS using PSL

For several reasons, we believe PSL is a more appropriate probabilistic logic for STS than MLNs. First, it is explicitly designed to support efficient inference, therefore it scales better to longer sentences with more complex logical forms. Second, it is also specifically designed for computing *similarity* between complex structured objects rather than determining probabilistic logical entailment. In fact, the initial version of PSL (Broecheler, Mihalkova, & Getoor, 2010) was called *Probabilistic Similarity Logic*, based on its use of *similarity functions*. This initial version was shown to be very effective for measuring the similarity of noisy database records and performing *record linkage* (i.e. identifying database entries referring to the same entity, such as bibliographic citations referring to the same paper).

This section explains how we adapt PSL's inference to be more suitable for the STS task (Beltagy et al., 2014a). For the same reason explained in section 3.3.2 that the conjunction tends to be more restrictive than required by the STS task, PSL does not work very well "out of the box". We show how to relax this conjunction, and make the required changes to the optimization problem and the grounding technique.

Changing Conjunction As mentioned above, Lukasiewicz's formula for conjunction is very restrictive and does not work well for STS. Therefore, we replace it with a new averaging interpretation of conjunction that we use to interpret the query Q. The truth value of the proposed average function is defined as:

$$I(p_1 \wedge .... \wedge p_n) = \frac{1}{n}\sum_{i=1}^{n} I(p_i) \tag{21}$$

where p_i is one of the conjuncted ground atoms. This averaging function is linear, and the result is a valid truth value in the interval [0, 1], therefore this change is easily incorporated into PSL without changing the complexity of inference which remains a linear-programming problem.

Heuristic Grounding Grounding is the process of instantiating the variables in the quantified rules with concrete constants in order to construct the nodes and links in the final graphical model. In principle, grounding requires instantiating each rule in all possible ways, substituting every possible constant for each variable in the rule. However, this is a combinatorial process that can easily result in an explosion in the size of the final network (same problem in MLN). Therefore, PSL employs a "lazy" approach to grounding that avoids the construction of irrelevant groundings. If there is no evidence for one of the antecedents in a particular grounding of a rule, then the normal PSL formula for conjunction guarantees that the rule is trivially satisfied ($I(r) = 1$) since the truth value of the antecedent is zero. Therefore, its distance to satisfaction is also zero, and it can be omitted from the ground network without impacting the result of MPE inference. This approach has similar effect as the modified closed-world assumption used with MLN in section 3.3.1.2.

However, this technique does not work once we switch to using averaging to interpret the query. For example, given the rule $\forall x.\, p(x) \wedge q(x) \Rightarrow t()$ and only one piece of evidence $p(C)$ there are no relevant groundings because there is no evidence for $q(C)$, and therefore, for normal PSL, $I(p(C) \wedge q(C)) = 0$ which does not affect $I(t())$. However, when using averaging with the same evidence, we need to generate the grounding $p(C) \wedge q(C)$ because $I(p(C) \wedge q(C)) = 0.5$ which *does* affect $I(t())$.

One way to solve this problem is to eliminate lazy grounding and generate all possible groundings. However, this produces an intractably large network. Therefore, we developed a heuristic approximate grounding technique that generates a subset of the most impactful groundings. Pseudocode for this heuristic approach is shown in algorithm 2. Its goal is to find constants that participate in ground atoms with high truth value and preferentially use them to construct a limited number of groundings of the query rule.

The algorithm takes the antecedents of a rule (in this case, the query formula Q) employing averaging conjunction as input. It also takes the *grounding limit* which is a threshold on the number of groundings to be returned. The algorithm uses several subroutines, they are:

- $Ant(v_i)$: given a variable v_i, it returns the set of rule antecedent atoms containing v_i. E.g, for the rule: $a(x) \wedge b(y) \wedge c(x)$, $Ant(x)$ returns the set of atoms $\{a(x), c(x)\}$.
- $Const(v_i)$: given a variable v_i, it returns the list of possible constants that can be used to instantiate the variable v_i.
- $Gnd(a_i)$: given an atom a_i, it returns the set of all possible ground atoms generated for a_i.
- $GndConst(a, g, v)$: given an atom a and grounding g for a, and a variable v, it finds the constant that substitutes for v in g. E.g, assume there is an atom $a = a_i(v_1, v_2)$, and the ground atom $g = a_i(A, B)$ is one of its groundings. $GndConst(a, g, v_2)$ would return the constant B since it is the substitution for the variable v_2 in g.

Algorithm 2 Heuristic Grounding

```
Input $r_{body} = a_1 \wedge .... \wedge a_n$: antecedent of a rule with average interpretation of conjunction
Input $V$: set of variables used in $r_{body}$
Input $Ant(v_i)$: subset of antecedents $a_j$ containing variable $v_i$
Input $Const(v_i)$: list of possible constants of variable $v_i$
Input $Gnd(a_i)$: set of ground atoms of $a_i$.
Input $GndConst(a, g, v)$: takes an atom $a$, grounding $g$ for $a$, and variable $v$, and returns the constant that
    substitutes $v$ in $g$
Input $gnd\_limit$: limit on the number of groundings
1: for all $v_i \in V$ do
2:     for all $C \in Const(v_i)$ do
3:         $score(C) = \sum_{a \in Ant(v_i)} (max\ I(g))$ for $g \in Gnd(a) \wedge GndConst(a, g, v_i) = C$
4:     end for
5:     sort $Const(v_i)$ on scores, descending
6: end for
7: return For all $v_i \in V$, take the Cartesian-product of the sorted $Const(v_i)$ and return the top $gnd\_limit$
    results
```

Lines 1-6 loop over all variables in the rule. For each variable, lines 2-5 construct a list of constants for that variable and sort it based on a heuristic score. In line 3, each constant is assigned a score that indicates the importance of this constant in terms of its impact on the truth value of the overall grounding. A constant's score is the sum, over all antecedents that contain the variable in question, of the maximum truth value of any grounding of that antecedent that contains that constant. Pushing constants with high scores to the top of each variable's list will tend to make the overall truth value of the top groundings high. Line 7 computes a subset of the Cartesian product of the sorted lists of constants, selecting constants in ranked order and limiting the number of results to the grounding limit.

One point that needs to be clarified about this approach is how it relies on the truth values of ground atoms when the goal of inference is to actually find these values. PSL's inference is actually an iterative process where in each iteration a grounding phase is followed by an optimization phase (solving the linear program). This loop repeats until convergence, i.e. until the truth values stop changing. The truth values used in each grounding phase come from the previous optimization phase. The first grounding phase assumes only the ground atoms in the evidence set have non-zero truth values.

3.4 Evaluation

This section presents the results of the evaluation of our semantic representation on the RTE and STS tasks. It starts with a description of the used datasets, then evaluation of the effect of the knowledge base, then evaluation of the inference step.

3.4.1 Datasets

We use three datasets for evaluation on the RTE and STS tasks

- **SICK(for RTE and STS):** "Sentences Involving Compositional Knowledge" (SICK) (Marelli et al., 2014) is a dataset collected for the SemEval 2014 competition. The dataset is 5,000 pairs for training and 5,000 for testing. Pairs are annotated for RTE and STS tasks.

	Task	RTE	STS		
	Dataset	SICK	SICK	msr-vid	msr-par
	dist	60.00 %	0.65	0.78	0.24
	state of the art	84.57 %	0.82	0.87	0.68
MLN	logic	73.44%	–	–	–
	logic+kb	**77.72%**	0.47	0.63	0.16
PSL	logic	n/a	0.72	0.74	0.46
	logic+kb	n/a	**0.74**	**0.79**	**0.53**

Table 1: System's performance, Accuracy for the RTE task, and Pearson Correlation for the STS task

- **msr-vid (for STS):** Microsoft Video Paraphrase Corpus from SemEval 2012 (Agirre et al., 2012) The dataset consists of 1,500 pairs of short video descriptions collected using crowdsourcing (Chen & Dolan, 2011) and subsequently annotated for the STS task. Half of the dataset is for training, and the second half is for testing.
- **msr-par (for STS):** Microsoft Paraphrase Corpus from SemEval 2012 (Agirre et al., 2012). The dataset is 5,801 pairs of sentences collected from news sources (Dolan, Quirk, & Brockett, 2004). Then, for STS 2012, 1,500 pairs were selected and annotated with similarity scores. Half of the dataset is for training, and the second half is for testing.

3.4.2 Knowledge Base Evaluation

This section evaluates our semantic representation compared to two baselines, distributional-only baseline and logic-only baseline.

Systems Compared

- **dist**: We use vector addition (Landauer & Dumais, 1997) as a distributional-only baseline. We compute a vector representation for each sentence by adding the distributional vectors of all of its words and measure similarity using cosine. This is a simple yet powerful baseline that uses only distributional information.
- **logic**: this is our probabilistic logic semantic representation but with no knowledge base.
- **logic+kb**: this is our probabilistic logic semantic representation with the knowledge base we build in section 3.2.

Results and Discussion Table 1 summarizes results of evaluating our semantic representation. RTE's performance is measured in Accuracy, and STS's performance is measured in Pearson correlation. For the RTE task, our MLN system (**logic** and **logic-kb**) out-performs the purely distributional baselines **dist**, because MLN benefits from the precision that the logic provides. For the STS task, our PSL system also out-performs the purely distributional baselines because it is able to combine the information available to **dist** in a better way that takes sentence structure into account. However, our MLN system for the STS task does not do as well as PSL. One reason is that MLN's performance is sensitive to the parameters of the average combiner, that is why we propose using weight learning to learn these parameters (section 4.4). Another reason is that MLN's inference for STS is very slow, and it times out in large number of pairs as we show in section 3.4.3. Table 1 also shows that adding the knowledge base, enhances the system performance. Inference rules are effectively representing the background knowledge, and allowing the inference to make better conclusions.

Table 1 also shows that our system is not performing as good as the state of the art. One major difference between our system and the top performing systems (Bär, Biemann, Gurevych, & Zesch, 2012; Lai & Hockenmaier, 2014; Zhu & Lan, 2014) is that they are large ensembles of simple features that are carefully engineered to the particular details of the datasets used for evaluation. For example, most of the contradicting RTE pairs in the SICK dataset are constructed using a simple negation like "T: A man is drawing", "H: There is no man drawing". This means that a simple feature that detects the existence of a negation operator is enough to correctly capture most of the contradicting pairs. On the other hand, our semantic representation is a general one that can be applied to any dataset, as we are not making any assumptions about the sentences (except that the parser can parse them). It can also be applied to more tasks other than RTE and STS as we discuss in the future work.

Error analysis can help us direct our future work to improve the performance of our system. This is an error analysis for our system's performance on the RTE task on the SICK dataset. From the confusion matrix, we find that the 22.28% misclassifications are distributed as follows,

- Entailment pairs classified as Neutral: 15.32%
- Contradiction pairs classified as Neutral: 6.12%
- Other: 0.84 %

This gives our system precision of 98.9% and recall of 78.56%. This is the typical behaviour of logic-base systems, that they have very high precision, but low recall. As concluded in (Bos, 2013), the low recall is mainly because of lack of enough knowledge base. Adding more inference rules and background information from different sources can help bridge this gap, as we explain in the future work. Also in the detection of contradiction, we found some limitations that we are also explaining and proposing a solution for, in the future work.

3.4.3 Inference Evaluation

This section evaluates the inference techniques based on accuracy and computational efficiency.

3.4.3.1 RTE Inference This is an evaluation for the different components of the inference process for the RTE task, namely the Query formula, and the Modified Closed-world.

Systems Compared

- **mln**: This system uses MC-SAT (Richardson & Domingos, 2006) for inference without any modifications. It uses the work-around explained in section 3.3.1.1 to calculate the probability of a complex query formula, and uses an open-world assumption.
- **mln+qf**: This system uses our SampleSearch inference to directly calculate the probability of a query formula (qf), while making an open-world assumption.
- **mln+mcw**: This system uses MC-SAT with the work-around for computing the probability of a complex query formula, but uses our modified closed-world (mcw) assumption.

	Accuracy	CPU Time	Timeouts
mln	56.94%	2min 27s	95.78%
mln+qf	68.74%	1min 51s	29.64%
mln+mcw	65.80%	10s	2.52%
mln+qf+mcw	71.80%	7s	2.12%

Table 2: Systems' performance, accuracy, CPU Time for completed runs only, and percentage of Timeouts

- **mln+qf+mcw**: This is our proposed technique, inference that supports a query formula (qf) and makes a modified closed-world (mcw) assumption.

We use a 30 minute timeout for each MLN inference run in order to make the experiments tractable. If the system times out, it outputs -1 indicating an error, and the classifier learns to assign it to one of the three RTE classes. Usually, because the Neutral class is the largest, timeouts are classified as Neutral.

Metrics

- Accuracy: Percentage of correct classifications (Entail, Contradict, or Neutral)
- CPU Time (completed runs): Average CPU time per run for the completed runs only, i.e. timed out runs are not included.
- Timeouts: Percentage of inferences that timeout after 30 minutes.

Results and Discussion Table 2 summarizes the results of the experiments. First, for all systems, the CPU time (average time per run for completed runs only) is very short compared to the length of the timeout (30 minutes). This shows the exponential nature of the inference algorithms, either the problem is small enough to finish in few minutes, or if it is slightly larger, it fails to finish in reasonable time.

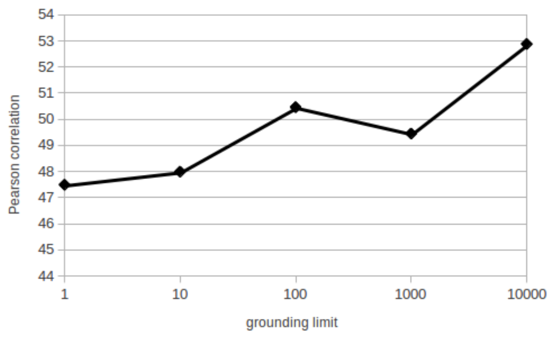
Comparing the systems, results clearly show that the base system, **mln**, is not effective for the type of inference problems that we are addressing, almost all of the runs timed out. System **mln+qf** shows the impact of being able to calculate the probability of a complex query directly. It significantly improves the accuracy, and it lowers the number of timeouts; however, the number of timeouts is still large. System **mln+mcw** shows the impact of the modified closed-world assumption, demonstrating that it makes inference significantly faster, since the number of unreachable ground atoms in our application is large compared to the total number of ground atoms. However, the accuracy of **mln+mcw** is lower than that of **mln+qf**, since calculating the probability of a query directly is more accurate than the standard work-around. Finally, **mln+qf+mcw** is both more accurate and faster than the other systems, clearly demonstrating the effectiveness of our overall approach.

3.4.3.2 STS Inference This section compares the computational efficiency of STS inferences on MLN and PSL.

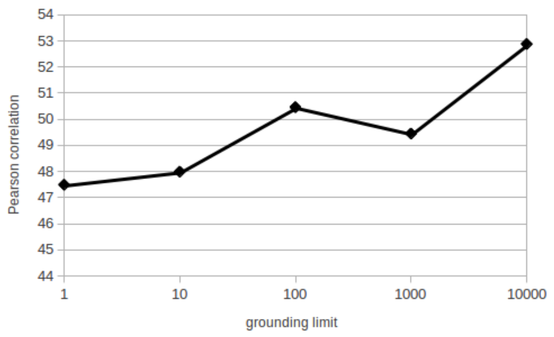
Computational Efficiency Table 3 shows the average CPU time for PSL and MLN inferences for the STS task. Because MLN's inference is slow, we use a timeout of 10 minutes. The results clearly demonstrate that PSL is an order of magnitude faster than MLN, and MLN's inference frequently times out. This is one

	PSL	MLN	
	CPU time	CPU time	timeouts
msr-vid	8s	1m 31s	8.8%
msr-par	30s	11m 49s	97.1%
SICK	10s	4m 24s	35.82%

Table 3: Average CPU time per STS pair, and percentage of timed-out pairs in MLN with a 10 minute time limit. PSL's grounding limit is set to 10,000 groundings.



(a) correlation score



(b) CPU time

Figure 3: Effect of PSL's grounding limit on performance for the **msr-par** dataset

of the reasons why the accuracy of MLN on the STS task is low as shown in table 1. This confirms that PSL is a better fit for the STS task.

As an attempt to improve MLN for the STS task, it should be possible to apply the modified closed-world assumption (section 3.3.1.2) that we developed for the RTE task on the STS task. This has the potential to significantly reduce the size of the problem and make MLN inference for STS faster. Applying the modified closed-world assumption to MLN also makes the comparison with PSL more fair, because PSL is already enforcing a comparable technique (lazy grounding) that plays a similar role in reducing the problem size.

PSL grounding limit We also evaluate the effect of changing the grounding limit on both Pearson correlation and CPU time for the **msr-par** dataset. Most of the sentences in **msr-par** are long, which results is large number of groundings, and limiting the number of groundings has a visible effect on the overall performance. In the other two datasets, the sentences are fairly short, and the full number of groundings is not large; therefore, changing the grounding limit does not significantly affect the results.

Figures 3a and 3b show the effect of changing the grounding limit on Pearson correlation and CPU time. As expected, as the grounding limit increases, accuracy improves but CPU time also increases. However, note that the difference in scores between the smallest and largest grounding limit tested is not large, suggesting that the heuristic approach to limit groundings is quite effective.

4 Proposed Research

This section discusses the proposed short-term research directions organized by the system component, followed by the long-term goals.

4.1 Parsing and Task Representation

Better RTE task formulation In the RTE task, we identify contradictions with the help of the inference $P(H|\neg T, KB)$. The intuition behind this additional inference is that if $\neg T \models H$, then T contradicts H. Although this helps detecting a lot of the contradictions in our experiments, it is not the best way to do so, because it misses many cases of contradictions. For example, consider this contradicting RTE pair: "T: No man is playing a flute", "H: A man is playing a large flute", which in logic are:

$$\begin{aligned} T&: \neg\exists x, y, z.\, man(x) \land agent(y, x) \land play(y) \land patient(y, z) \land flute(z) \\ \neg T&: \exists x, y, z.\, man(x) \land agent(y, x) \land play(y) \land patient(y, z) \land flute(z) \\ H&: \exists x, y, z.\, man(x) \land agent(y, x) \land play(y) \land patient(y, z) \land large(z) \land flute(z) \end{aligned}$$

It is clear from the example that $\neg T \not\models H$, and we get a wrong conclusion.

Logically, detection of contradiction is checking that $T \land H \models False$ which is equivalent to $T \models \neg H$. However, the probabilistic counterpart of $T \models \neg H$ is $P(\neg H|T)$ which equals $1 - P(H|T)$, and that means evaluating $P(\neg H|T)$ does not add any extra information other than what is provided by $P(H|T)$, and it can not be used to detect contradictions. The solution comes from the fact that $T \models \neg H$ is logically equivalent to $H \models \neg T$, but its probabilistic counterpart $P(\neg T|H)$ does not equal $P(\neg H|T)$. This suggests that the inference that we need for the detection of contradictions is $P(\neg T|H)$ because it is logically correct, and it is probabilistically more informative than $P(\neg H|T)$.

With the enhancement discussed above, we will be detecting the Entailments and Contradictions using the two inferences $P(H|T)$ and $P(\neg T|H)$. A better way to detect them would be to take the ratios $\frac{P(H|T)}{P(H)}$ and $\frac{P(\neg T|H)}{P(\neg T)}$. The intuition behind the ratios is that they measure to what extent adding the evidence changes the probability of the query from its prior probability. For example, large values for $\frac{P(H|T)}{P(H)}$ means that adding T increases probability of H which is a stronger indication of Entailment than just the value of $P(H|T)$. Similarly, values around 1 mean that T does not affect probability of H which is an indication that T is Neutral to H, and values close to 0 are another indicator of Contradiction.

DCA and Negated Existential We discussed in section 3.1.2 how to get correct inferences for universally quantified hypothesis H despite the finite domain restriction that DCA enforces. We make sure that a universally quantified H is entailed not just because it is true for all entities in the domain (as enforced by DCA), but also because of an explicit universal quantification in T. We only supported one form of universal quantifiers, but we do not have support for the negated existential form of universal quantifiers, e.g.

$$\neg\exists x, y.\, young(x) \land girl(x) \land agent(y, x) \land dance(y) \tag{22}$$

In finite domains, and because of the closed-world assumption that enforces everything to be false by default, H could come to be true no matter what T says. However, we need H to be true only if T is explicitly negating the existence of a young girl that dances. One possible way to achieve that goal is to add to the

MLN a rule R representing the negated part of H, and set its weight to a high value, but not infinity. This way, without T, H will have a very low probability. H can not be true unless T (which has infinite weight) is explicitly negating R. Here is an RTE example adapted from the SICK dataset, "T: A young girl is standing on one leg", "H: There is no young girl dancing" which in logic are:

T: $\exists x, y, z.\, young(x) \wedge girl(x) \wedge agent(y, x) \wedge stand(y) \wedge patient(y, z) \wedge one(z) \wedge leg(z)$
H: $\neg\exists x, y.\, young(x) \wedge girl(x) \wedge agent(y, x) \wedge dance(y)$
R: $young(G) \wedge girl(G) \wedge agent(D, G) \wedge dance(D) | w = 5.0$

For the detection of entailment, we need to compute $P(T|H)$. We want $P(T|H)$ to be 0, but we actually get $P(T|H) = 1$ because by default, the young girl is not dancing. This is an undesired inference because T is not explicitly negating the dancing. Then we generate R from the negated part of H. $P(H|T, R) \simeq 0$ and that is because T is not explicitly negating R, which is the correct inference we need to conclude that T does not entail H.

4.2 Knowledge Base Construction

One of the advantages of using a probabilistic logic is that additional sources of rules can easily be incorporated by adding additional soft inference rules. We propose adding two more types of rules

Paraphrase Rules In addition to WordNet, we can add rules from explicit paraphrase collections like the ones by Berant, Dagan, and Goldberger (2011), and PPDB (Ganitkevitch et al., 2013). These are precompiled collections of rules, not generated on-the-fly as we do with the distributional rules. To be able to use these rules in our system, they need to be translated into logic, then weighted. For example, the paraphrase rule "solve"⇒"find a solution to" should be translated to:

$$\forall e, x.\, solve(e) \wedge patient(e, x) \Rightarrow \exists s.\, find(e) \wedge patient(e, s) \wedge solution(s) \wedge to(t, x) \tag{23}$$

The tricky part is how to match variables of the left-hand side into the predicates on the right-hand side of the rule, we call this step *variable binding*. For simple rules that do not have many entities on each side of the rule, we can define a set of "templates" or patterns for them. For example, variable binding of a rule like noun-phrase⇒noun-phrase is simple because each side has only one variable. Templates can handle most of the rules, but not all of them. For more complex cases, we are planning to convert to logic both the natural language sentence and the sentence after the rule has been applied to it, then extract the rule in first-order logic from there. After translating the rules to logic, the rule's weight that comes with the paraphrase collection need to be mapped to a probabilistic logic weight. There are different possible ways to map paraphrase weights to probabilistic logic weights. One way is to use an equation similar to the one used with the distributional semantics as shown in section 3.2.2, and this assumes that paraphrase weights can be normalized to values between zero and one. Another way is to use weight learning as we discuss in the future work in section 4.4

Phrasal Distributional Rules In addition to the lexical distributional rules, we plan to generate phrasal distributional rules. Phrasal distributional rules are inference rules generated between short phrases (not individual words). Weights of the rules come from distributional semantics. The rules will be generated based on linguistically motivated "templates". A template specifies what a phrase is, and how variables are mapped between the left-hand side and the right-hand side of the rule (variable binding). The simplest template is "noun-phrase ⇒ noun-phrase", e.g. $\forall x.\, little(x) \wedge kid(x) \Rightarrow smart(x) \wedge boy(x)$. It is the simplest template because each side of the rule has a single variable, and the variable binding is trivial. A more complex rule could be between verb phrases, "subject-noun-phrase + verb + object-noun-phrase ⇒ subject-noun-phrase + verb + object-noun-phrase". Each side has three variables, and variable binding is to map subject to subject, verb to verb and object to object, e.g.

$$\begin{aligned} \forall x, y, z.\, man(x) \wedge agent(y, x) \wedge drive(y) \wedge patient(y, z) \wedge car(z) \\ \Rightarrow guy(x) \wedge agent(y, x) \wedge ride(y) \wedge patient(y, z) \wedge bike(z) \end{aligned} \tag{24}$$

Templates are defined based on our linguistic knowledge, and based on the capabilities of distributional semantics. It is better to avoid using phrases that distributional semantics can not efficiently represent as vectors. Different distributional compositionality techniques can be tried, from simple ones like vector addition and component-wise multiplication (Mitchell & Lapata, 2008, 2010) to more sophisticated ones (Grefenstette & Sadrzadeh, 2011; Paperno, Pham, & Baroni, 2014).

4.3 Inference

Better inference for MLN with Query Formula Our MLN inference algorithm to calculate the probability of a query discussed in section 3.3.1.1 can be enhanced. Instead of making two separate runs of SampleSearch to estimate two different Zs, it would be helpful to exploit the similarities between the two Markov networks (one with Q and one without Q) to reduce the amount of repeated computation. Also, it should be possible to optimize the calculations, or simplify them, knowing that we are really only interested in the ratio between the two Zs and not their individual values.

Generalized Modified Closed-World assumption Our algorithm for the modified closed-world assumption works only for the current form of inference rules we generate. It assumes that all inference rules are of the form $\forall v_1..v_n \; lhs \Rightarrow rhs$ where lhs and rhs are sets of conjuncted predicates. This assumptions simplifies the implementation of propagation of evidence from the lhs of a rule to its rhs. However, this technique is not general enough to handle more complex forms of rules. For example, it is not clear how to propagate evidence for a rule like: $\forall x.\, short(x) \Leftrightarrow \neg long(x)$. By default, all entities are "not short" and "not long", which contradicts this rule, and there is no obvious way to decide what entities to be "short" and what entities to be"long". We need a general technique that for arbitrary MLNs, it can decide what ground atoms have their probabilities change during the inference process, and what ground atoms remain having their prior probabilities.

4.4 Learning

All the work we have done so far was in inference, but probabilistic logic frameworks also support learning for the weights. Weight learning can be applied to our system in many ways, some of them are listed below. We would like to attempt at least one of them.

- Weights that we have on inference rules, either distributional or precompiled paraphrases, are learned from large collection of text, but they are not learned specifically for being used as weights of inference rules in probabilistic logic. Although the function f in equation 15 plays an important role to map these weights to MLN weights, it would be more accurate to learn this mapping from the training set. Because of limited training data, we can not learn a weight per a rule. Instead, we can learn a weight per a rule type. We can think of the current function f as a prior, then extend it with a type-dependant parameter that we learn in the learning process.

- In the STS task, learn to assign different weights to different parts of the sentence where higher weight indicates that annotators pay more attention to that part. For example, we could learn that the type of an object determined by a noun should be weighted more than a property specified by an adjective. As a result, “black dog” would be appropriately judged more similar to “white dog” than to “black cat.”. Because the training data is limited, we can not learn different weights for different words. Instead, we can learn weights for different fragments of the sentence. Sentence fragments need to be abstracted to fragment type. A candidate fragment types is using the CCG Supertag of each word as its category. This means, we only need to learn a weight per a CCG Supertag. In MLN, weights are applied through weights of the rules of the average combiner. In PSL, weights are applied through the averaging equation which is replaced with weighted average.

4.5 Long Term

In the long-term, we propose to extend our work in the following directions:

Question Answering Our semantic representation is a deep flexible semantic representation that can be used to perform various types of tasks, not just RTE and STS. We are interested in applying our semantic representation to the question answering task. Question answering is the task of finding an answer of a WH question from large corpus of *unstructured* text. All the text is translated to logic, and the question is translated to a logical expression with existentially quantified variable representing the questioned part. Then probabilistic logic inference tool needs to find the best entities in the text that fill in that existential quantifier in the question. Existing logic-based systems are usually applied to limited domains, such as querying a specific database (Kwiatkowski et al., 2013; Berant et al., 2013), but with our system, we have the potential to query a large corpus of unstructured text because we are using Boxer for wide-coverage semantic analysis. The interesting bottleneck is the inference. It would be very challenging to scale probabilistic logic inference to such large inference problems.

A related, but not similar task is given the same corpus of text as in the question answering task and with *no* question, we would like to generating the n most probable inferences. In (Angeli & Manning, 2014), this task is named “Common Sense Reasoning”. Performing this task in probabilistic logic requires a new type of inference algorithms that can generate these inferences without a specific query to direct the search.

Generalized Quantifiers Generalized quantifiers are words like Few, Most, Many, Only .. etc (Barwise & Cooper, 1981). First-order logic has native support for two of them, Every and Some. Also, some of them can be encoded relatively easy in first-order logic, like Exactly. However, some of them like Few, Most and Many are not natively supported in first-order logic, and they can not be encoded easily. We would like to add support for these generalized quantifiers in our system. A generalized quantifier has a restrictor and body. In “Most birds fly”, the generalized quantifier “Most” has the restrictor “birds” and the body “fly”. One way to add support for generalized quantifiers in the RTE task, is to reason about the direction of entailments between the restrictors and the bodies of the sentences. For example, consider the RTE pair: “T: Most big birds fly high”, “H: Some birds fly”. We can conclude that T entails H because the restrictor “big birds” of T entails the restrictor “birds” of H, and the body “fly high” of T entails the body “fly” of H.

Another way to represent some of the generalized quantifiers, especially Most and Few, is to replace Most or Few with “Every” and give the rule a low weight indicating that some worlds could be violating it. Setting the weight is a function of the generalized quantifier being represented.

Contextualized WordNet rules We add WordNet rules with infinite weights. However, because words are ambiguous based on context, a better way to represent the WordNet information is to use Word Sense Disambiguation (WSD) (Navigli, 2009; Tan, 2014) to disambiguate words, then generate "weighted" rules for possible senses, where the WSD's confidence is mapped to a rule weight.

Other Languages we would like to see how our semantic representation can be applied on languages other than English. Theoretically, the proposed semantic representation is language independent, but practically, not all the resources and tools are available. Other than English, the only CCGbank (to train a CCG parser) available is Chinese CCGbank (Tse & Curran, 2010). As for Boxer, it only supports English, but there are future plans for its multi-language support. Once there is a CCGbank and Boxer available for a language, the rest of the pipeline is relatively easy. Building a vector space and generating lexical rules is straight forward. Also any available resource of background knowledge can be mapped to logical rules and added to the system. The inference part should be the same with no changes.

Inference Inspector Probabilistic logic tools need to be extended to output information about the inference process, what rules were useful to reach the conclusion and what rules were not. This can be a useful tool to debug the system and identify the problems. Such extra information is helpful for the RTE task to know which part of T is the reason to entail which part of H. This gives insights on how the inference process goes, and facilitates finding the kind of missing knowledge bases. BIUTEE (Stern & Dagan, 2013) is an example of an RTE system that adds explanations to the inference.

Implementing such tool in PSL would be easier than in MLN. PSL's inference is an optimization problem, so it should be enough to list the rules found at the critical point where the optimal solution lies. In MLN, it is not as straight forward as in PSL, because in MLN all rules affect the output to some extent. Some rules affect it dramatically, others slightly affect it. So what we need is being able to find the rules with the most impact.

5 Conclusions

Being able to effectively represent natural languages semantics is important and has a lot of potential applications. In this proposal, we use probabilistic logic to represent natural language semantics to combine the expressivity and the automated inference of the logical representations, with the ability to capture graded aspect of natural languages as in distributional semantics. We evaluate this semantic representation on two end tasks that require deep semantic understanding, RTE and STS. Our system maps natural sentences to logical formulas, use them to build probabilistic logic inference problems, build a knowledge base from precompiled resources and on-the-fly resources, then perform inference for the RTE and STS tasks on MLN and PSL using our inference algorithms that are tailored for the type of inference problems we are interested in. Experiments showed that our system can handle RTE and STS reasonably well.

For the future work, we enhance formulation of the RTE task, build bigger knowledge base from more resources, generalize our modified closed-world assumption, enhance our MLN inference algorithm, and use some weight learning. Our long term goals are applying our semantic representation to the question answering task, support generalized quantifiers, contextualizing our WordNet rules, applying our semantic representation to other languages and implementing a probabilistic logic inference inspector that visualises the inference process.

6 Acknowledgments

This research was supported in part by the NSF CAREER grant IIS 0845925, by an NDSEG grant, by MURI ARO grant W911NF-08-1-0242 and by the DARPA DEFT program under AFRL grant FA8750-13-2-0026. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the view of DARPA, AFRL, ARO, DoD or the US government. Most of our experiments were run on the Mastodon Cluster supported by NSF Grant EIA-0303609.

References

Agirre, E., Cer, D., Diab, M., & Gonzalez-Agirre, A. (2012). SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval-2012)*.

Angeli, G., & Manning, C. D. (2014). Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*.

Bach, S. H., Huang, B., London, B., & Getoor, L. (2013). Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proceedings of 29th Conference on Uncertainty in Artificial Intelligence (UAI-2013)*.

Bär, D., Biemann, C., Gurevych, I., & Zesch, T. (2012). UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval-2012)*.

Baroni, M., & Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*.

Barwise, J., & Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, *4*(2), 159–219.

Beltagy, I., Chau, C., Boleda, G., Garrette, D., Erk, K., & Mooney, R. (2013). Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM-2013)*.

Beltagy, I., Erk, K., & Mooney, R. (2014a). Probabilistic soft logic for semantic textual similarity. In *Proceedings of Association for Computational Linguistics (ACL-2014)*.

Beltagy, I., Erk, K., & Mooney, R. (2014b). Semantic parsing using distributional semantics and probabilistic logic. In *Proceedings of ACL 2014 Workshop on Semantic Parsing (SP-2014)*.

Beltagy, I., & Mooney, R. J. (2014). Efficient Markov logic inference for natural language semantics. In *Proceedings of AAAI 2014 Workshop on Statistical Relational AI (StarAI-2014)*.

Beltagy, I., Roller, S., Boleda, G., Erk, K., & Mooney, R. J. (2014). UTexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*.

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*.

Berant, J., Dagan, I., & Goldberger, J. (2011). Global learning of typed entailment rules. In *Proceedings of Association for Computational Linguistics (ACL-2011)*.

Bos, J. (2008). Wide-coverage semantic analysis with Boxer. In *Proceedings of Semantics in Text Processing (STEP-2008)*.

Bos, J. (2013). Is there a place for logic in recognizing textual entailment?. *Linguistic Issues in Language Technology*, *9*.

Broecheler, M., Mihalkova, L., & Getoor, L. (2010). Probabilistic Similarity Logic. In *Proceedings of 26th Conference on Uncertainty in Artificial Intelligence (UAI-2010).*

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: a library for support vector machines.. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, D. L., & Dolan, W. B. (2011). Collecting highly parallel data for paraphrase evaluation. In *Proceedings of Association for Computational Linguistics (ACL-2011).*

Clark, S., & Curran, J. R. (2004). Parsing the WSJ using CCG and log-linear models. In *Proceedings of Association for Computational Linguistics (ACL-2004).*

Dagan, I., Roth, D., Sammons, M., & Zanzotto, F. M. (2013). Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, *6*(4), 1–220.

Dechter, R., Kask, K., & Mateescu, R. (2002). Iterative join-graph propagation. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002).*

Dolan, B., Quirk, C., & Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING-2004).*

Erk, K., & Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2008).*

Friedman, J. (1999). Stochastic gradient boosting. Tech. rep., Stanford University.

Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2013). PPDB: The paraphrase database. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013).*

Garrette, D., Erk, K., & Mooney, R. (2011). Integrating logical representations with probabilistic information using Markov logic. In *Proceedings of International Conference on Computational Semantics (IWCS-2011).*

Getoor, L., & Taskar, B. (Eds.). (2007). *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Gogate, V. (2014). IJGP-sampling and SampleSearch.. Software available at `http://www.hlt.utdallas.edu/~vgogate/ijgp-samplesearch.html`.

Gogate, V., & Dechter, R. (2011). SampleSearch: Importance sampling in presence of determinism. *Artificial Intelligence*, *175*(2), 694–729.

Gogate, V., & Domingos, P. (2011). Probabilistic theorem proving. In *Proceedings of 27th Conference on Uncertainty in Artificial Intelligence (UAI-2011).*

Grefenstette, E. (2013). Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of Second Joint Conference on Lexical and Computational Semantics (*SEM-2013).*

Grefenstette, E., & Sadrzadeh, M. (2011). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2011).*

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, *11*(1), 10–18.

Kamp, H., & Reyle, U. (1993). *From Discourse to Logic*. Kluwer.

Kimmig, A., Bach, S. H., Broecheler, M., Huang, B., & Getoor, L. (2012). A short introduction to Probabilistic Soft Logic. In *Proceedings of NIPS Workshop on Probabilistic Programming: Foundations and Applications (NIPS Workshop-2012)*.

Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). The Alchemy system for statistical relational AI.. `http://www.cs.washington.edu/ai/alchemy`.

Kwiatkowski, T., Choi, E., Artzi, Y., & Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*.

Lai, A., & Hockenmaier, J. (2014). Illinois-lh: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*.

Landauer, T., & Dumais, S. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, *104*(2), 211.

Lewis, M., & Steedman, M. (2013). Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics (TACL-2013)*, *1*, 179–192.

Lin, D., & Pantel, P. (2001). Discovery of inference rules for question answering. *Natural Language Engineering*, *7*(4), 343–360.

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, *28*(2), 203–208.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., & Zamparelli, R. (2014). A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of Association for Computational Linguistics (ACL-2008)*.

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, *34*(3), 1388–1429.

Montague, R. (1970). Universal grammar. *Theoria*, *36*, 373–398.

Natarajan, S., Khot, T., Lowd, D., Tadepalli, P., Kersting, K., & Shavlik, J. (2010). Exploiting causal independence in Markov logic networks: Combining undirected and directed models. In *Proceedings of European Conference in Machine Learning (ECML-2010)*.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, *41*(2), 10.

Paperno, D., Pham, N. T., & Baroni, M. (2014). A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of Association for Computational Linguistics (ACL-2014)*.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, Boston, MA.

Princeton University (2010). About WordNet.. `http://wordnet.princeton.edu`.

Raina, R., Ng, A. Y., & Manning, C. D. (2005). Robust textual inference via learning and abductive reasoning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Roller, S., Erk, K., & Boleda, G. (2014). Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the Twenty Fifth International Conference on Computational Linguistics (COLING-2014)*.

Schutze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123.

Singla, P., & Domingos, P. (2008). Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*.

Stern, A., & Dagan, I. (2013). The BIUTEE research platform for transformation-based textual entailment recognition. In *Linguistics Issues in Language Technology (LiLT-2013)*.

Szpektor, I., & Dagan, I. (2008). Learning entailment rules for unary templates. In *Proceedings of the Twenty Second International Conference on Computational Linguistics (COLING-2008)*.

Tan, L. (2014). Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software].. `https://github.com/alvations/pywsd`.

Thater, S., Fürstenau, H., & Pinkal, M. (2010). Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of Association for Computational Linguistics (ACL-2010)*.

Thomason, R. H. (Ed.). (1974). *Formal Philosophy. Selected Papers of Richard Montague*. Yale University Press, New Haven.

Tse, D., & Curran, J. R. (2010). Chinese CCGbank: extracting CCG derivations from the Penn Chinese Treebank. In *Proceedings of the Twenty Third International Conference on Computational Linguistics (COLING-2010)*.

Turney, P., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, *37*(1), 141–188.

Venugopal, D., & Gogate, V. (2013). GiSS: Combining SampleSearch and Importance Sampling for inference in mixed probabilistic and deterministic graphical models. In *Proceedings of Association for the Advancement of Artificial Intelligence(AAAI-13)*.

Zhu, T., & Lan, M. (2014). ECNU: Leveraging on ensemble of heterogeneous features and information enrichment for cross level semantic similarity estimation. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*.