Probabilistic Soft Logic for Semantic Textual Similarity

Islam Beltagy[§] **Katrin Erk**[†] **Raymond Mooney**[§]
[§]Department of Computer Science
[†]Department of Linguistics
The University of Texas at Austin
Austin, Texas 78712
[§]{beltagy,mooney}@cs.utexas.edu
[†]katrin.erk@mail.utexas.edu

Abstract

Probabilistic Soft Logic (PSL) is a recently developed framework for probabilistic logic. We use PSL to combine logical and distributional representations of natural-language meaning, where distributional information is represented in the form of weighted inference rules. We apply this framework to the task of Semantic Textual Similarity (STS) (i.e. judging the semantic similarity of natural-language sentences), and show that PSL gives improved results compared to a previous approach based on Markov Logic Networks (MLNs) and a purely distributional approach.

1 Introduction

When will people say that two sentences are similar? This question is at the heart of the Semantic Textual Similarity task (STS)(Agirre et al., 2012). Certainly, if two sentences contain many of the same words, or many similar words, that is a good indication of sentence similarity. But that can be misleading. A better characterization would be to say that if two sentences use the same or similar words *in the same or similar relations*, then those two sentences will be judged similar.[1] Interestingly, this characterization echoes the principle of compositionality, which states that the meaning of a phrase is uniquely determined by the meaning of its parts and the rules that connect those parts.

Beltagy et al. (2013) proposed a hybrid approach to sentence similarity: They use a very deep representation of sentence meaning, expressed in first-order logic, to capture sentence structure, but combine it with distributional similarity ratings at the word and phrase level. Sentence similarity is then modelled as mutual entailment in a probabilistic logic. This approach is interesting in that it uses a very deep and precise representation of meaning, which can then be relaxed in a controlled fashion using distributional similarity. But the approach faces large hurdles in practice, stemming from efficiency issues with the Markov Logic Networks (MLN) (Richardson and Domingos, 2006) that they use for performing probabilistic logical inference.

In this paper, we use the same combined logic-based and distributional framework as Beltagy et al., (2013) but replace Markov Logic Networks with Probabilistic Soft Logic (PSL) (Kimmig et al., 2012; Bach et al., 2013). PSL is a probabilistic logic framework designed to have efficient inference. Inference in MLNs is theoretically intractable in the general case, and existing approximate inference algorithms are computationally expensive and sometimes inaccurate. Consequently, the MLN approach of Beltagy et al. (2013) was unable to scale to long sentences and was only tested on the relatively short sentences in the Microsoft video description corpus used for STS (Agirre et al., 2012). On the other hand, inference in PSL reduces to a linear programming problem, which is theoretically and practically much more efficient. Empirical results on a range of problems have confirmed that inference in PSL is much more efficient than in MLNs, and frequently more accurate (Kimmig et al., 2012; Bach et al., 2013).

We show how to use PSL for STS, and describe changes to the PSL framework that make it more effective for STS. For evaluation, we test on three STS datasets, and compare our PSL system with the MLN approach of Beltagy et al., (2013) and with distributional-only baselines. Experimental

[1] Mitchell and Lapata (2008) give an amusing example of two sentences that consist of all the same words, but are very different in their meaning: (a) It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem. (b) That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1210–1219,
Baltimore, Maryland, USA, June 23-25 2014.

results demonstrate that, overall, PSL models human similarity judgements more accurately than these alternative approaches, and is significantly faster than MLNs.

The rest of the paper is organized as follows: section 2 presents relevant background material, section 3 explains how we adapted PSL for the STS task, section 4 presents the evaluation, and sections 5 and 6 discuss future work and conclusions, respectively.

2 Background

2.1 Logical Semantics

Logic-based representations of meaning have a long tradition (Montague, 1970; Kamp and Reyle, 1993). They handle many complex semantic phenomena such as relational propositions, logical operators, and quantifiers; however, their binary nature 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 semantically parsing sentences into logical expressions utilizing such an ontology is very difficult. Consequently, current semantic parsers are mostly restricted to quite limited domains, such as querying a specific database (Kwiatkowski et al., 2013; Berant et al., 2013). In contrast, our system is not limited to any formal ontology and can use a wide-coverage tool for semantic analysis, as discussed below.

2.2 Distributional Semantics

Distributional models (Turney and Pantel, 2010), on the other hand, use statistics on contextual data from large corpora to predict semantic similarity of words and phrases (Landauer and Dumais, 1997; Mitchell and Lapata, 2010). They are relatively easier to build than logical representations, automatically acquire knowledge from "big data," and capture the "graded" nature of linguistic meaning, but do not adequately capture logical structure (Grefenstette, 2013).

Distributional models 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 and Dumais, 1997; Lund and 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 and Dumais, 1997) or by a component-wise product of word vectors (Mitchell and Lapata, 2008; Mitchell and Lapata, 2010), or more complex methods that compute phrase vectors from word vectors and tensors (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011). We use vector addition (Landauer and Dumais, 1997), and component-wise product (Mitchell and Lapata, 2008) as baselines for STS. Vector addition was previously found to be the best performing simple distributional method for STS (Beltagy et al., 2013).

2.3 Markov Logic Networks

Markov Logic Networks (MLN) (Richardson and Domingos, 2006) are a framework for probabilistic logic that employ weighted formulas in first-order logic to compactly encode complex undirected probabilistic graphical models (i.e., Markov networks). 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. 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. A variety of inference methods for MLNs have been developed, however, developing a scalable, general-purpose, accurate inference method for complex MLNs is an open problem. Beltagy et al. (2013) use MLNs to represent the meaning of natural language sentences and judge textual entailment and semantic similarity, but they were unable to scale the approach beyond short sentences due to the complexity of MLN inference.

2.4 Probabilistic Soft Logic

Probabilistic Soft Logic (PSL) is a recently proposed alternative framework for probabilistic logic (Kimmig et al., 2012; Bach et al., 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:

$$\forall x, y, z.\, friend(x, y) \wedge votesFor(y, z) \rightarrow votesFor(x, z) \mid 0.3 \quad (1)$$

$$\forall x, y, z.\, spouse(x, y) \wedge votesFor(y, z) \rightarrow votesFor(x, z) \mid 0.8 \quad (2)$$

In our notation, we use lower case letters like x, y, z to represent variables and upper case letters for constants. 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, in our application, we generate inference rules that incorporate the similarity of two predicates. This can be represented in PSL as:

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

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:

$$I(l_1 \wedge l_1) = max\{0, I(l_1) + I(l_2) - 1\}$$
$$I(l_1 \vee l_1) = min\{I(l_1) + I(l_2), 1\}$$
$$I(\neg l_1) = 1 - I(l_1)$$

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 (2). 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:

$$p(I) = \frac{1}{Z} \exp\,[-\sum_{r \in R} \lambda_r (d(r))^p]; \quad (3)$$

$$Z = \int_I \exp\,[-\sum_{r \in R} \lambda_r (d(r))^p]$$

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 3, 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).

2.5 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. The best performing system in 2012's competition was by Bär et al. (2012), a complex ensemble system that integrates many techniques including string similarity, n-gram overlap, WordNet similarity, vector space similarity and MT evaluation metrics. Two of the datasets we use for evaluation are from the 2012 competition. We did not utilize the new datasets added in the 2013 competition since they did not contain naturally-occurring, full sentences, which is the focus of our work.

2.6 Combining logical and distributional methods using probabilistic logic

There are a few recent attempts to combine logical and distributional representations in order to obtain the advantages of both. 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 that is important to measuring semantic similarity.

Garrette et al. (2011) introduced a framework for combining logic and distributional models using probabilistic logic. Distributional similarity between pairs of words is converted into weighted inference rules that are added to the logical representation, and Markov Logic Networks are used to perform probabilistic logical inference.

Beltagy et al. (2013) extended this framework by generating distributional inference rules from phrase similarity and tailoring the system to the STS task. STS is treated as computing the probability of two textual entailments $T \models H$ and $H \models T$, where T and H are the two sentences whose similarity is being judged. These two entailment probabilities are averaged to produce a measure of similarity. The MLN constructed to determine the probability of a given entailment includes the logical forms for both T and H as well as soft inference rules that are constructed from distributional information. Given a similarity score for all pairs of sentences in the dataset, a regressor is trained on the training set to map the system's output to the gold standard scores. The trained regressor is applied to the scores in the test set before calculating Pearson correlation. The regression algorithm used is Additive Regression (Friedman, 2002).

To determine an entailment probability, first, the two sentences are mapped to logical representations using Boxer (Bos, 2008), a tool for wide-coverage semantic analysis that maps a CCG (Combinatory Categorial Grammar) parse into a lexically-based logical form. Boxer uses C&C for CCG parsing (Clark and Curran, 2004).

Distributional semantic knowledge is then encoded as weighted inference rules in the MLN. A rule's weight (w) is a function of the cosine similarity (sim) between its antecedent and consequent. Rules are generated on the fly for each T and H. Let t and h be the lists of all words and phrases in T and H respectively. For all pairs (a, b), where $a \in t, b \in h$, it generates an inference rule: $a \rightarrow b \mid w$, where $w = f(sim(\overrightarrow{a}, \overrightarrow{b}))$. Both a and b can be words or phrases. Phrases are defined in terms of Boxer's output. A phrase is more than one unary atom sharing the same variable like "a little kid" which in logic is $little(K) \wedge kid(K)$. A phrase also can be two unary atoms connected by a relation like "a man is driving" which in logic is $man(M) \wedge agent(D, M) \wedge drive(D)$. The similarity function sim takes two vectors as input. Phrasal vectors are constructed using Vector Addition (Landauer and Dumais, 1997). The set of generated inference rules can be regarded as the knowledge base KB.

Beltagy et al. (2013) found that the logical conjunction in H is very restrictive for the STS task, so they relaxed the conjunction by using an average evidence combiner (Natarajan et al., 2010). The average combiner results in computationally complex inference and only works for short sentences. In case inference breaks or times-out, they back off to a simpler combiner that leads to much faster inference but loses most of the structure of the sentence and is therefore less accurate.

Given T, KB and H from the previous steps, MLN inference is then used to compute $p(H|T, KB)$, which is then used as a measure of the degree to which T entails H.

3 PSL for STS

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 was also specifically designed for computing *similarity* between complex structured objects rather than determining probabilistic logical entailment. In fact, the initial version of PSL (Broecheler et al., 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). Therefore, we have developed an approach that follows that of Beltagy et al. (2013), but replaces Markov Logic with PSL.

This section explains how we formulate the STS task as a PSL program. PSL does not work very well "out of the box" for STS, mainly because Lukasiewicz's equation for the conjunction is very restrictive. Therefore, we use a different interpretation for conjunction that uses averaging, which requires corresponding changes to the optimization problem and the grounding technique.

3.1 Representation

Given the logical forms for a pair of sentences, a text T and a hypothesis H, and given a set of weighted rules derived from the distributional semantics (as explained in section 2.6) composing the knowledge base KB, we build a PSL model that supports determining the truth value of H in the most probable interpretation (i.e. MPE) given T and KB.

Consider the pair of sentences is "A man is driving", and "A guy is walking". Parsing into logical form gives:

$$T:\ \exists x,y.\ man(x) \wedge agent(y,x) \wedge drive(y)$$

$$H:\ \exists x,y.\ guy(x) \wedge agent(y,x) \wedge walk(y)$$

The PSL program is constructed as follows:

***T*:** The text is represented in the evidence set. For the example, after Skolemizing the existential quantifiers, this contains the ground atoms: $\{man(A), agent(B,A), drive(B)\}$

***KB*:** The knowledge base is a set of lexical and phrasal rules generated from distributional semantics, along with a similarity score for each rule (section 2.6). For the example, we generate the rules: $\forall x.\ man(x) \wedge vs\_sim(\text{"}man\text{"}, \text{"}guy\text{"}) \rightarrow guy(x)$, $\forall x. drive(x) \wedge vs\_sim(\text{"}drive\text{"}, \text{"}walk\text{"}) \rightarrow walk(x)$

where vs_sim is a similarity function that calculates the distributional similarity score between the two lexical predicates. All rules are assigned the same weight because all rules are equally important.

***H*:** The hypothesis is represented as $H \rightarrow result()$, and then PSL is queried for the truth value of the atom $result()$. For our example, the rule is: $\forall x,y.\ guy(x) \wedge agent(y,x) \wedge walk(y) \rightarrow result()$.

Priors: A low prior is given to all predicates. This encourages the truth values of ground atoms to be zero, unless there is evidence to the contrary.

For each STS pair of sentences S_1, S_2, we run PSL twice, once where $T = S_1$, $H = S_2$ and another where $T = S_2$, $H = S_1$, and output the two scores. To produce a final similarity score, we train a regressor to learn the mapping between the two PSL scores and the overall similarity score. As in Beltagy et al., (2013) we use Additive Regression (Friedman, 2002).

3.2 Changing Conjunction

As mentioned above, Lukasiewicz's formula for conjunction is very restrictive and does not work well for STS. For example, for T: "A man is driving" and H: "A man is driving a car", if we use the standard PSL formula for conjunction, the output value is zero because there is no evidence for a car and $max(0, X + 0 - 1) = 0$ for any truth value $0 \leq X \leq 1$. However, humans find these sentences to be quite similar.

Therefore, we introduce a new averaging interpretation of conjunction that we use for the hypothesis H. The truth value for a conjunction is defined as $I(p_1 \wedge .... \wedge p_n) = \frac{1}{n}\sum_{i=1}^{n} I(p_i)$. 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.

It would perhaps be even better to use a weighted average, where weights for different components are learned from a supervised training set. This is an important direction for future work.

3.3 Grounding Process

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. 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

Algorithm 1 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
```

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.

However, this technique does not work once we switch to using averaging to interpret conjunctions. 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 1. Its goal is to find constants that participate in ground propositions with high truth value and preferentially use them to construct a limited number of groundings of each rule.

The algorithm takes the antecedents of a rule 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.

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 antedents 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 propositions in the evidence provided have non-zero truth values.

4 Evaluation

This section evaluates the performance of PSL on the STS task.

4.1 Datasets

We evaluate our system on three STS datasets.

- **msr-vid**: Microsoft Video Paraphrase Corpus from STS 2012. The dataset consists of 1,500 pairs of short video descriptions collected using crowdsourcing (Chen and Dolan, 2011) and subsequently annotated for the STS task (Agirre et al., 2012). Half of the dataset is for training, and the second half is for testing.

- **msr-par**: Microsoft Paraphrase Corpus from STS 2012 task. The dataset is 5,801 pairs of sentences collected from news sources (Dolan et al., 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.

- **SICK**: Sentences Involving Compositional Knowledge is a dataset collected for SemEval 2014. Only the training set is available at this point, which consists of 5,000 pairs of sentences. Pairs are annotated for RTE and STS, but we only use the STS data. Training and testing was done using 10-fold cross validation.

4.2 Systems Compared

We compare our PSL system with several others. In all cases, we use the distributional word vectors employed by Beltagy et al. (2013) based on context windows from Gigaword.

- **vec-add**: Vector Addition (Landauer and Dumais, 1997). 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.

- **vec-mul**: Component-wise Vector Multiplication (Mitchell and Lapata, 2008). The same as **vec-add** except uses component-wise multiplication to combine word vectors.

- **MLN**: The system of Beltagy et al. (2013), which uses Markov logic instead of PSL for probabilistic inference. MLN inference is very slow in some cases, so we use a 10 minute timeout. When MLN times out, it backs off to a simpler sentence representation as explained in section 2.6.

- **PSL**: Our proposed PSL system for combining logical and distributional information.

- **PSL-no-DIR**: Our PSL system without distributional inference rules(empty knowledge base). This system uses PSL to compute similarity of logical forms but does not use distributional information on lexical or phrasal similarity. It tests the impact of the probabilistic logic only

- **PSL+vec-add**: **PSL** ensembled with **vec-add**. Ensembling the MLN approach with a purely distributional approach was found to improve results (Beltagy et al., 2013), so we also tried this with PSL. The methods are ensembled by using both entailment scores of both systems as input features to the regression step that learns to map entailment scores to STS similarity ratings. This way, the training data is used to learn how to weight the contribution of the different components.

- **PSL+MLN**: **PSL** ensembled with **MLN** in the same manner.

4.3 Experiments

Systems are evaluated on two metrics, Pearson correlation and average CPU time per pair of sentences.

- Pearson correlation: The Pearson correlation between the system's similarity scores and the human gold-standards.

- CPU time: This metric only applies to MLN and PSL. The CPU time taken by the inference step is recorded and averaged over all pairs in each of the test datasets. In many cases, MLN inference is very slow, so we timeout after 10 minutes and report the number of timed-out pairs on each dataset.

	msr-vid	**msr-par**	**SICK**
vec-add	0.78	0.24	0.65
vec-mul	0.76	0.12	0.62
MLN	0.63	0.16	0.47
PSL-no-DIR	0.74	0.46	0.68
PSL	0.79	**0.53**	0.70
PSL+vec-add	**0.83**	0.49	**0.71**
PSL+MLN	0.79	0.51	0.70
Best Score (Bär et al., 2012)	0.87	0.68	n/a

Table 1: STS Pearson Correlations

	PSL	**MLN**	
	time	time	timeouts/total
msr-vid	8s	1m 31s	132/1500
msr-par	30s	11m 49s	1457/1500
SICK	10s	4m 24s	1791/5000

Table 2: Average CPU time per STS pair, and number of timed-out pairs in MLN with a 10 minute time limit. PSL's grounding limit is set to 10,000 groundings.

We also evaluated 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.

4.4 Results and Discussion

Table 1 shows the results for Pearson correlation. **PSL** out-performs the purely distributional baselines (**vec-add** and **vec-mul**) because **PSL** is able to combine the information available to **vec-add** and **vec-mul** in a better way that takes sentence structure into account. **PSL** also outperforms the unaided probabilistic-logic baseline that does not use distributional information (**PSL-no-DIR**). **PSL-no-DIR** works fairly well because there is significant overlap in the exact words and structure of the paired sentences in the test data, and PSL combines the evidence from these similarities effectively. In addition, **PSL** always does significantly better than **MLN**, because of the large number of timeouts, and because the conjunction-averaging in PSL is combining evidence better than MLN's average-combiner, whose performance is sensitive to various parameters. These results further support the claim that using probabilistic logic to integrate logical and distributional information is a promising approach to natural-language semantics. More specifically, they strongly indicate that PSL is a more effective probabilistic logic for judging semantic similarity than MLNs.

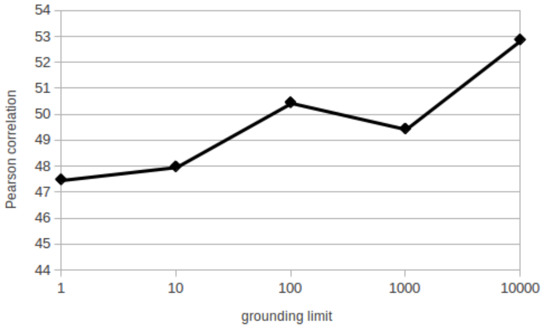

Figure 1: Effect of **PSL**'s grounding limit on the correlation score for the **msr-par** dataset

Like for MLNs (Beltagy et al., 2013), ensembling PSL with vector addition improved the scores a bit, except for **msr-par** where **vec-add**'s performance is particularly low. However, this ensemble still does not beat the state of the art (Bär et al., 2012) which is a large ensemble of many different systems. It would be informative to add our system to their ensemble to see if it could improve it even further.

Table 2 shows the CPU time for **PSL** and **MLN**. The results clearly demonstrate that **PSL** is an order of magnitude faster than **MLN**.

Figures 1 and 2 show the effect of changing the grounding limit on Pearson correlation and CPU time. As expected, as the grounding limit is increased, 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 limiting grounding is quite effective.

5 Future Work

As mentioned in Section 3.2, it would be good to use a *weighted* average to compute the truth

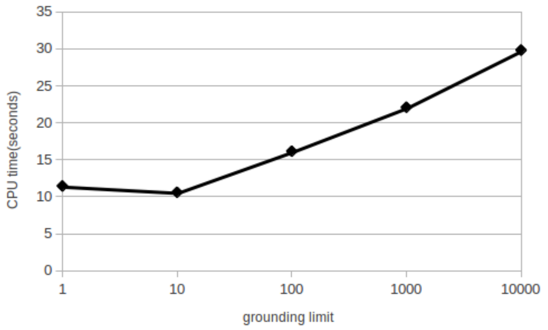

Figure 2: Effect of **PSL**'s grounding limit on CPU time for the **msr-par** dataset

values for conjunctions, weighting some predicates more than others rather than treating them all equally. Appropriate weights for different components could be learned from training data. For example, such an approach 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."

One of the advantages of using a probabilistic logic is that additional sources of knowledge can easily be incorporated by adding additional soft inference rules. To complement the soft inference rules capturing distributional lexical and phrasal similarities, PSL rules could be added that encode explicit paraphrase rules, such as those mined from monolingual text (Berant et al., 2011) or multi-lingual parallel text (Ganitkevitch et al., 2013).

This paper has focused on STS; however, as shown by Beltagy et al. (2013), probabilistic logic is also an effective approach to *recognizing textual entailment* (RTE). By using the appropriate functions to combine truth values for various logical connectives, PSL could also be adapted for RTE. Although we have shown that PSL outperforms MLNs on STS, we hypothesize that MLNs may still be a better approach for RTE. However, it would be good to experimentally confirm this intuition. In any case, the high computational complexity of MLN inference could mean that PSL is still a more practical choice for RTE.

6 Conclusion

This paper has presented an approach that uses Probabilistic Soft Logic (PSL) to determine Semantic Textual Similarity (STS). The approach uses PSL to effectively combine logical semantic representations of sentences with soft inference rules for lexical and phrasal similarities computed from distributional information. The approach builds upon a previous method that uses Markov Logic (MLNs) for STS, but replaces the probabilistic logic with PSL in order to improve the efficiency and accuracy of probabilistic inference. The PSL approach was experimentally evaluated on three STS datasets and was shown to outperform purely distributional baselines as well as the MLN approach. The PSL approach was also shown to be much more scalable and efficient than using MLNs

Acknowledgments

This research was supported 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, DoD or the US government. Some experiments were run on the Mastodon Cluster supported by NSF Grant EIA-0303609.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of Semantic Evaluation (SemEval-12)*.

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proceedings of Uncertainty in Artificial Intelligence (UAI-13)*.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of Semantic Evaluation (SemEval-12)*.

Marco Baroni and Roberto Zamparelli. 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-10)*.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013.

Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM-13)*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of Association for Computational Linguistics (ACL-11)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*.

Johan Bos. 2008. Wide-coverage semantic analysis with Boxer. In *Proceedings of Semantics in Text Processing (STEP-08)*.

Matthias Broecheler, Lilyana Mihalkova, and Lise Getoor. 2010. Probabilistic Similarity Logic. In *Proceedings of Uncertainty in Artificial Intelligence (UAI-20)*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of Association for Computational Linguistics (ACL-11)*.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of Association for Computational Linguistics (ACL-04)*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the International Conference on Computational Linguistics (COLING-04)*.

Jerome H Friedman. 2002. Stochastic gradient boosting. *Journal of Computational Statistics & Data Analysis (CSDA-02)*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-13)*.

Dan Garrette, Katrin Erk, and Raymond Mooney. 2011. Integrating logical representations with probabilistic information using Markov logic. In *Proceedings of International Conference on Computational Semantics (IWCS-11)*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*.

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

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer.

Angelika Kimmig, Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A short introduction to Probabilistic Soft Logic. In *Proceedings of NIPS Workshop on Probabilistic Programming: Foundations and Applications (NIPS Workshop-12)*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*.

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

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics (TACL-13)*.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of Association for Computational Linguistics (ACL-08)*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Journal of Cognitive Science*.

Richard Montague. 1970. Universal grammar. *Theoria*, 36:373–398.

Sriraam Natarajan, Tushar Khot, Daniel Lowd, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. 2010. Exploiting causal independence in Markov logic networks: Combining undirected and directed models. In *Proceedings of European Conference in Machine Learning (ECML-10)*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research (JAIR-10)*.