Semantic Frame Identification with Distributed Word Representations

Karl Moritz Hermann[‡*] **Dipanjan Das**[†] **Jason Weston**[†] **Kuzman Ganchev**[†]
[‡]Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom
[†] Google Inc., 76 9th Avenue, New York, NY 10011, United States
karl.moritz.hermann@cs.ox.ac.uk
{dipanjand,kuzman}@google.com jaseweston@gmail.com

Abstract

We present a novel technique for semantic frame identification using distributed representations of predicates and their syntactic context; this technique leverages automatic syntactic parses and a generic set of word embeddings. Given labeled data annotated with frame-semantic parses, we learn a model that projects the set of word representations for the syntactic context around a predicate to a low dimensional representation. The latter is used for semantic frame identification; with a standard argument identification method inspired by prior work, we achieve state-of-the-art results on FrameNet-style frame-semantic analysis. Additionally, we report strong results on PropBank-style semantic role labeling in comparison to prior work.

1 Introduction

Distributed representations of words have proved useful for a number of tasks. By providing richer representations of meaning than what can be encompassed in a discrete representation, such approaches have successfully been applied to tasks such as sentiment analysis (Socher et al., 2011), topic classification (Klementiev et al., 2012) or word-word similarity (Mitchell and Lapata, 2008).

We present a new technique for semantic **frame** identification that leverages distributed word representations. According to the theory of frame semantics (Fillmore, 1982), a semantic frame represents an event or scenario, and possesses frame elements (or semantic **roles**) that participate in the event. Most work on frame-semantic parsing has usually divided the task into two major subtasks: *frame identification*, namely the disambiguation of a given predicate to a frame, and *argument identification* (or semantic role labeling), the analysis of words and phrases in the sentential context that satisfy the frame's semantic roles (Das et al., 2010; Das et al., 2014).[1] Here, we focus on the first subtask of frame identification for given predicates; we use our novel method (§3) in conjunction with a standard argument identification model (§4) to perform full frame-semantic parsing.

We present experiments on two tasks. First, we show that for frame identification on the FrameNet corpus (Baker et al., 1998; Fillmore et al., 2003), we outperform the prior state of the art (Das et al., 2014). Moreover, for full frame-semantic parsing, with the presented frame identification technique followed by our argument identification method, we report the best results on this task to date. Second, we present results on PropBank-style semantic role labeling (Palmer et al., 2005; Meyers et al., 2004; Màrquez et al., 2008), that approach strong baselines, and are on par with prior state of the art (Punyakanok et al., 2008).

2 Overview

Early work in frame-semantic analysis was pioneered by Gildea and Jurafsky (2002). Subsequent work in this area focused on either the FrameNet or PropBank frameworks, and research on the latter has been more popular. Since the CoNLL 2004-2005 shared tasks (Carreras and Màrquez,

[*]The majority of this research was carried out during an internship at Google.

[1]There are exceptions, wherein the task has been modeled using a pipeline of three classifiers that perform frame identification, a binary stage that classifies candidate arguments, and argument identification on the filtered candidates (Baker et al., 2007; Johansson and Nugues, 2007).

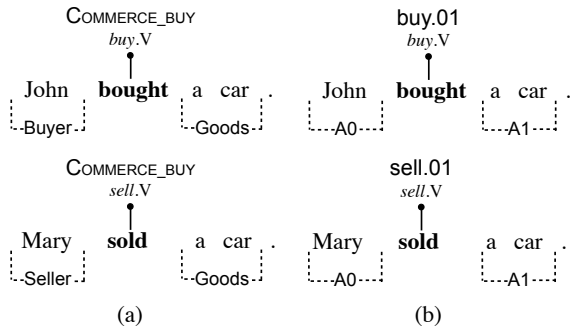

Figure 1: Example sentences with frame-semantic analyses. FrameNet annotation conventions are used in (a) while (b) denotes PropBank conventions.

2004; Carreras and Màrquez, 2005) on PropBank semantic role labeling (SRL), it has been treated as an important NLP problem. However, research has mostly focused on argument analysis, skipping the frame disambiguation step, and its interaction with argument identification.

2.1 Frame-Semantic Parsing

Closely related to SRL, frame-semantic parsing consists of the resolution of predicate sense into a frame, and the analysis of the frame's arguments. Work in this area exclusively uses the FrameNet full text annotations. Johansson and Nugues (2007) presented the best performing system at SemEval 2007 (Baker et al., 2007), and Das et al. (2010) improved performance, and later set the current state of the art on this task (Das et al., 2014). We briefly discuss FrameNet, and subsequently PropBank annotation conventions here.

FrameNet The FrameNet project (Baker et al., 1998) is a lexical database that contains information about words and phrases (represented as lemmas conjoined with a coarse part-of-speech tag) termed as lexical units, with a set of semantic frames that they could evoke. For each frame, there is a list of associated frame elements (or roles, henceforth), that are also distinguished as core or non-core.[2] Sentences are annotated using this universal frame inventory. For example, consider the pair of sentences in Figure 1(a). COMMERCE_BUY is a frame that can be evoked by morphological variants of the two example lexical units *buy*.V and *sell*.V. Buyer, Seller and Goods are some example roles for this frame.

[2]Additional information such as finer distinction of the coreness properties of roles, the relationship between frames, and that of roles are also present, but we do not leverage that information in this work.

PropBank The PropBank project (Palmer et al., 2005) is another popular resource related to semantic role labeling. The PropBank corpus has verbs annotated with sense frames and their arguments. Like FrameNet, it also has a lexical database that stores type information about verbs, in the form of sense frames and the possible semantic roles each frame could take. There are modifier roles that are shared across verb frames, somewhat similar to the non-core roles in FrameNet. Figure 1(b) shows annotations for two verbs "bought" and "sold", with their lemmas (akin to the lexical units in FrameNet) and their verb frames buy.01 and sell.01. Generic core role labels (of which there are seven, namely A0-A5 and AA) for the verb frames are marked in the figure.[3] A key difference between the two annotation systems is that PropBank uses a local frame inventory, where frames are predicate-specific. Moreover, role labels, although few in number, take specific meaning for each verb frame. Figure 1 highlights this difference: while both *sell*.V and *buy*.V are members of the same frame in FrameNet, they evoke different frames in PropBank. In spite of this difference, nearly identical statistical models could be employed for both frameworks.

Modeling In this paper, we model the frame-semantic parsing problem in two stages: **frame identification** and **argument identification**. As mentioned in §1, these correspond to a frame disambiguation stage,[4] and a stage that finds the various arguments that fulfill the frame's semantic roles within the sentence, respectively. This resembles the framework of Das et al. (2014), who solely focus on FrameNet corpora, unlike this paper. The novelty of this paper lies in the frame identification stage (§3). Note that this two-stage approach is unusual for the PropBank corpora when compared to prior work, where the vast majority of published papers have not focused on the verb frame disambiguation problem at all, only focusing on the role labeling stage (see the overview paper of Màrquez et al. (2008) for example).

[3]NomBank (Meyers et al., 2004) is a similar resource for nominal predicates, but we do not consider it in our experiments.

[4]For example in PropBank, the lexical unit *buy*.V has three verb frames and in sentential context, we want to disambiguate its frame. (Although PropBank never formally uses the term lexical unit, we adopt its usage from the frame semantics literature.)

2.2 Distributed Frame Identification

We present a model that takes word embeddings as input and learns to identify semantic frames. A word embedding is a distributed representation of meaning where each word is represented as a vector in $\mathbb{R}^n$. Such representations allow a model to share meaning between similar words, and have been used to capture semantic, syntactic and morphological content (Collobert and Weston, 2008; Turian et al., 2010, *inter alia*). We use word embeddings to represent the syntactic context of a particular predicate instance as a vector. For example, consider the sentence "He runs the company." The predicate *runs* has two syntactic dependents – a subject and direct object (but no prepositional phrases or clausal complements). We could represent the syntactic context of *runs* as a vector with blocks for all the possible dependents warranted by a syntactic parser; for example, we could assume that positions $0 \ldots n$ in the vector correspond to the subject dependent, $n{+}1 \ldots 2n$ correspond to the clausal complement dependent, and so forth. Thus, the context is a vector in $\mathbb{R}^{nk}$ with the embedding of *He* at the subject position, the embedding of *company* in direct object position and zeros everywhere else. Given input vectors of this form for our training data, we learn a matrix that maps this high dimensional and sparse representation into a lower dimensional space. Simultaneously, the model learns an embedding for all the possible labels (i.e. the frames in a given lexicon). At inference time, the predicate-context is mapped to the low dimensional space, and we choose the nearest frame label as our classification. We next describe this model in detail.

3 Frame Identification with Embeddings

We continue using the example sentence from §2.2: "He runs the company." where we want to disambiguate the frame of *runs* in context. First, we extract the words in the syntactic context of *runs*; next, we concatenate their word embeddings as described in §2.2 to create an initial vector space representation. Subsequently, we learn a mapping from this initial representation into a low-dimensional space; we also learn an embedding for each possible frame label in the same low-dimensional space. The goal of learning is to make sure that the correct frame label is as close as possible to the mapped context, while competing frame labels are farther away.

Formally, let x represent the actual sentence with a marked predicate, along with the associated syntactic parse tree; let our initial representation of the predicate context be $g(x)$. Suppose that the word embeddings we start with are of dimension n. Then g is a function from a parsed sentence x to $\mathbb{R}^{nk}$, where k is the number of possible syntactic context types. For example g selects some important positions relative to the predicate, and reserves a block in its output space for the embedding of words found at that position. Suppose g considers clausal complements and direct objects. Then $g : X \rightarrow \mathbb{R}^{2n}$ and for the example sentence it has zeros in positions $0 \ldots n$ and the embedding of the word *company* in positions $n{+}1 \ldots 2n$.

$$g(x) = [0, \ldots, 0, \text{embedding of } \textit{company}].$$

Section 3.1 describes the context positions we use in our experiments. Let the low dimensional space we map to be $\mathbb{R}^m$ and the learned mapping be $M : \mathbb{R}^{nk} \rightarrow \mathbb{R}^m$. The mapping M is a linear transformation, and we learn it using the WSABIE algorithm (Weston et al., 2011). WSABIE also learns an embedding for each frame label (y, henceforth). In our setting, this means that each frame corresponds to a point in $\mathbb{R}^m$. If we have F possible frames we can store those parameters in an $F \times m$ matrix, one m-dimensional point for each frame, which we will refer to as the linear mapping Y. Let the lexical unit (the lemma conjoined with a coarse POS tag) for the marked predicate be ℓ. We denote the frames that associate with ℓ in the frame lexicon[5] and our training corpus as F_ℓ. WSABIE performs gradient-based updates on an objective that tries to minimize the distance between $M(g(x))$ and the embedding of the correct label $Y(y)$, while maintaining a large distance between $M(g(x))$ and the other possible labels $Y(\bar{y})$ in the confusion set F_ℓ. At disambiguation time, we use a simple dot product similarity as our distance metric, meaning that the model chooses a label by computing the $\text{argmax}_y s(x, y)$ where $s(x, y) = M(g(x)) \cdot Y(y)$, where the argmax iterates over the possible frames $y \in F_\ell$ if ℓ was seen in the lexicon or the training data, or $y \in F$, if it was unseen.[6] Model learning is performed using the margin ranking loss function as described in

[5]The frame lexicon stores the frames, corresponding semantic roles and the lexical units associated with the frame.

[6]This disambiguation scheme is similar to the one adopted by Das et al. (2014), but they use unlemmatized words to define their confusion set.

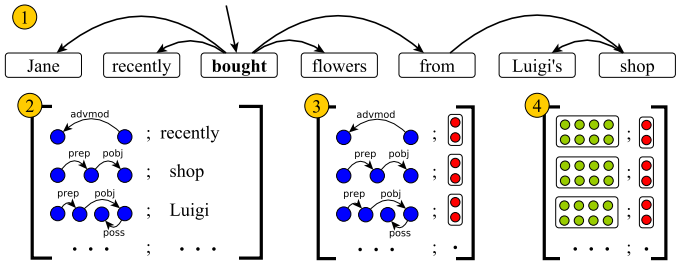

Figure 2: Context representation extraction for the embedding model. Given a dependency parse (1) the model extracts all words matching a set of paths from the frame evoking predicate and its direct dependents (2). The model computes a composed representation of the predicate instance by using distributed vector representations for words (3) – the (red) vertical embedding vectors for each word are concatenated into a long vector. Finally, we learn a linear transformation function parametrized by the context blocks (4).

Weston et al. (2011), and in more detail in section 3.2.

Since WSABIE learns a single mapping from $g(x)$ to $\mathbb{R}^m$, parameters are shared between different words and different frames. So for example "He *runs* the company" could help the model disambiguate "He *owns* the company." Moreover, since $g(x)$ relies on word embeddings rather than word identities, information is shared between words. For example "*He* runs *the company*" could help us to learn about "*She* runs *a corporation*".

3.1 Context Representation Extraction

In principle $g(x)$ could be any feature function, but we performed an initial investigation of two particular variants. In both variants, our representation is a block vector where each block corresponds to a syntactic position relative to the predicate, and each block's values correspond to the embedding of the word at that position.

Direct Dependents The first context function we considered corresponds to the examples in §3. To elaborate, the positions of interest are the labels of the direct dependents of the predicate, so k is the number of labels that the dependency parser can produce. For example, if the label on the edge between *runs* and *He* is nsubj, we would put the embedding of *He* in the block corresponding to nsubj. If a label occurs multiple times, then the embeddings of the words below this label are averaged.

Unfortunately, using only the direct dependents can miss a lot of useful information. For example, topicalization can place discriminating information farther from the predicate. Consider "He *runs* the *company*." vs. "It was the *company* that he *runs*." In the second sentence, the discriminating word, *company* dominates the predicate *runs*. Similarly, predicates in embedded clauses may have a distant agent which cannot be captured using direct dependents. Consider "*The athlete ran* the marathon." vs. "*The athlete* prepared himself for three months to *run* the marathon." In the second example, for the predicate *run*, the agent *The athlete* is not a direct dependent, but is connected via a longer dependency path.

Dependency Paths To capture more relevant context, we developed a second context function as follows. We scanned the training data for a given task (either the PropBank or the FrameNet domains) for the dependency paths that connected the gold predicates to the gold semantic arguments. This set of dependency paths were deemed as possible positions in the initial vector space representation. In addition, akin to the first context function, we also added all dependency labels to the context set. Thus for this context function, the block cardinality k was the sum of the number of scanned gold dependency path types and the number of dependency labels. Given a predicate in its sentential context, we therefore extract only those context words that appear in positions warranted by the above set. See Figure 2 for an illustration of this process.

We performed initial experiments using context extracted from 1) direct dependents, 2) dependency paths, and 3) both. For all our experiments, setting 3) which concatenates the direct dependents and dependency path always dominated the other two, so we only report results for this setting.

3.2 Learning

We model our objective function following Weston et al. (2011), using a weighted approximate-rank pairwise loss, learned with stochastic gradient descent. The mapping from $g(x)$ to the low dimensional space $\mathbb{R}^m$ is a linear transformation, so the model parameters to be learnt are the matrix $M \in \mathbb{R}^{nk \times m}$ as well as the embedding of each possible frame label, represented as another matrix $Y \in \mathbb{R}^{F \times m}$ where there are F frames in total. The training objective function minimizes:

$$\sum_x \sum_{\bar{y}} L\big(rank_y(x)\big) \max(0, \gamma + s(x,y) - s(x,\bar{y})).$$

where x, y are the training inputs and their corresponding correct frames, and $\bar{y}$ are negative frames, γ is the margin. Here, $rank_y(x)$ is the rank of the positive frame y relative to all the negative frames:

$$rank_y(x) = \sum_{\bar{y}} I(s(x,y) \leq \gamma + s(x,\bar{y})),$$

and $L(\eta)$ converts the rank to a weight. Choosing $L(\eta) = C\eta$ for any positive constant C optimizes the mean rank, whereas a weighting such as $L(\eta) = \sum_{i=1}^{\eta} 1/i$ (adopted here) optimizes the top of the ranked list, as described in (Usunier et al., 2009). To train with such an objective, stochastic gradient is employed. For speed the computation of $rank_y(x)$ is then replaced with a sampled approximation: sample N items $\bar{y}$ until a violation is found, i.e. $\max(0, \gamma + s(x,\bar{y}) - s(x,y))) > 0$ and then approximate the rank with $(F-1)/N$, see Weston et al. (2011) for more details on this procedure. For the choices of the stochastic gradient learning rate, margin (γ) and dimensionality (m), please refer to §5.4-§5.5.

Note that an alternative approach could learn only the matrix M, and then use a k-nearest neighbor classifier in $\mathbb{R}^m$, as in Weinberger and Saul (2009). The advantage of learning an embedding for the frame labels is that at inference time we need to consider only the set of labels for classification rather than all training examples. Additionally, since we use a frame lexicon that gives us the possible frames for a given predicate, we usually only consider a handful of candidate labels. If we used all training examples for a given predicate for finding a nearest-neighbor match at inference time, we would have to consider many more candidates, making the process very slow.

4 Argument Identification

Here, we briefly describe the argument identification model used in our frame-semantic parsing experiments, post frame identification. Given x, the sentence with a marked predicate, the argument identification model assumes that the predicate frame y has been disambiguated. From a frame lexicon, we look up the set of semantic roles $\mathcal{R}_y$ that associate with y. This set also contains the null role $r_\emptyset$. From x, a rule-based candidate argument extraction algorithm extracts a set of spans $\mathcal{A}$ that could potentially serve as the overt[7] arguments $\mathcal{A}_y$ for y (see §5.4-§5.5 for the details of the candidate argument extraction algorithms).

[7]By overtness, we mean the non-null instantiation of a semantic role in a frame-semantic parse.

• starting word of a	• POS of the starting word of a
• ending word of a	• POS of the ending word of a
• head word of a	• POS of the head word of a
• bag of words in a	• bag of POS tags in a
• a bias feature	• voice of the predicate use
• word cluster of a's head	
• word cluster of a's head conjoined with word cluster of the predicate*	
• dependency path between a's head and the predicate	
• the set of dependency labels of the predicate's children	
• dependency path conjoined with the POS tag of a's head	
• dependency path conjoined with the word cluster of a's head	
• position of a with respect to the predicate (*before, after, overlap or identical*)	
• whether the subject of the predicate is missing (*missingsubj*)	
• *missingsubj*, conjoined with the dependency path	
• *missingsubj*, conjoined with the dependency path from the verb dominating the predicate to a's head	

Table 1: Argument identification features. The span in consideration is termed a. Every feature in this list has two versions, one conjoined with the given role r and the other conjoined with both r and the frame y. The feature with a * superscript is only conjoined with the role to reduce its sparsity.

Learning Given training data of the form $\langle\langle x^{(i)}, y^{(i)}, \mathcal{M}^{(i)}\rangle\rangle_{i=1}^{N}$, where,

$$\mathcal{M} = \{(r,a\} : r \in \mathcal{R}_y, a \in \mathcal{A} \cup \mathcal{A}_y\}, \quad (1)$$

a set of tuples that associates each role r in $\mathcal{R}_y$ with a span a according to the gold data. Note that this mapping associates spans with the null role $r_\emptyset$ as well. We optimize the following log-likelihood to train our model:

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{j=1}^{|\mathcal{M}^{(i)}|} \log p_{\boldsymbol{\theta}}\big((r,a)_j | x, y, \mathcal{R}_y\big) - C\|\boldsymbol{\theta}\|_2^2$$

where $p_{\boldsymbol{\theta}}$ is a log-linear model normalized over the set $\mathcal{R}_y$, with features described in Table 1. We set $C = 1.0$ and use L-BFGS (Liu and Nocedal, 1989) for training.

Inference Although our learning mechanism uses a local log-linear model, we perform inference globally on a per-frame basis by applying hard structural constraints. Following Das et al. (2014) and Punyakanok et al. (2008) we use the log-probability of the local classifiers as a score in an integer linear program (ILP) to assign roles subject to hard constraints described in §5.4 and §5.5. We use an off-the-shelf ILP solver for inference.

5 Experiments

In this section, we present our experiments and the results achieved. We evaluate our novel frame identification approach in isolation and also conjoined with argument identification resulting in full frame-semantic structures; before presenting our model's performance we first focus on the datasets, baselines and the experimental setup.

5.1 Data

We evaluate our models on both FrameNet- and PropBank-style structures. For FrameNet, we use the full-text annotations in the FrameNet 1.5 release[8] which was used by Das et al. (2014, §3.2). We used the same test set as Das et al. containing 23 documents with 4,458 predicates. Of the remaining 55 documents, 16 documents were randomly chosen for development.[9]

For experiments with PropBank, we used the Ontonotes corpus (Hovy et al., 2006), version 4.0, and only made use of the Wall Street Journal documents; we used sections 2-21 for training, section 24 for development and section 23 for testing. This resembles the setup used by Punyakanok et al. (2008). All the verb frame files in Ontonotes were used for creating our frame lexicon.

5.2 Frame Identification Baselines

For comparison, we implemented a set of baseline models, with varying feature configurations. The baselines use a log-linear model that models the following probability at training time:

$$p(y|x,\ell) = \frac{e^{\boldsymbol{\psi}\cdot\mathbf{f}(y,x,\ell)}}{\sum_{\bar{y}\in F_\ell} e^{\boldsymbol{\psi}\cdot\mathbf{f}(\bar{y},x,\ell)}} \quad (2)$$

At test time, this model chooses the best frame as $\text{argmax}_y \boldsymbol{\psi}\cdot\mathbf{f}(y,x,\ell)$ where argmax iterates over the possible frames $y \in F_\ell$ if ℓ was seen in the lexicon or the training data, or $y \in F$, if it was unseen, like the disambiguation scheme of §3. We train this model by maximizing L_2 regularized log-likelihood, using L-BFGS; the regularization constant was set to 0.1 in all experiments.

For comparison with our model from §3, which we call WSABIE EMBEDDING, we implemented two baselines with the log-linear model. Both the baselines use features very similar to the input representations described in §3.1. The first one computes the direct dependents and dependency paths as described in §3.1 but conjoins them with the word identity rather than a word embedding. Additionally, this model uses the un-conjoined words as backoff features. This would be a standard NLP approach for the frame identification problem, but is surprisingly competitive with the state of the art. We call this baseline LOG-LINEAR WORDS. The second baseline, tries to decouple the WSABIE training from the embedding input, and trains a log linear model using the embeddings. So the second baseline has the same input representation as WSABIE EMBEDDING but uses a log-linear model instead of WSABIE. We call this model LOG-LINEAR EMBEDDING.

5.3 Common Experimental Setup

We process our PropBank and FrameNet training, development and test corpora with a shift-reduce dependency parser that uses the Stanford conventions (de Marneffe and Manning, 2013) and uses an arc-eager transition system with beam size of 8; the parser and its features are described by Zhang and Nivre (2011). Before parsing the data, it is tagged with a POS tagger trained with a conditional random field (Lafferty et al., 2001) with the following emission features: word, the word cluster, word suffixes of length 1, 2 and 3, capitalization, whether it has a hyphen, digit and punctuation. Beyond the bias transition feature, we have two cluster features for the left and right words in the transition. We use Brown clusters learned using the algorithm of Uszkoreit and Brants (2008) on a large English newswire corpus for cluster features. We use the same word clusters for the argument identification features in Table 1.

We learn the initial embedding representations for our frame identification model (§3) using a deep neural language model similar to the one proposed by Bengio et al. (2003). We use 3 hidden layers each with 1024 neurons and learn a 128-dimensional embedding from a large corpus containing over 100 billion tokens. In order to speed up learning, we use an unnormalized output layer and a hinge-loss objective. The objective tries to ensure that the correct word scores higher than a random incorrect word, and we train with mini-batch stochastic gradient descent.

5.4 Experimental Setup for FrameNet

Hyperparameters For our frame identification model with embeddings, we search for the WSABIE hyperparameters using the development data.

[8] See `https://framenet.icsi.berkeley.edu`.

[9] These documents are listed in appendix A.

		SEMAFOR LEXICON			FULL LEXICON		
	Model	All	Ambiguous	Rare	All	Ambiguous	Rare
Development Data	LOG-LINEAR WORDS	89.21	72.33	88.22	89.28	72.33	88.37
	LOG-LINEAR EMBEDDING	88.66	72.41	87.53	88.74	72.41	87.68
	WSABIE EMBEDDING (§3)	**90.78**	**76.43**	**90.18**	**90.90**	**76.83**	**90.18**

		SEMAFOR LEXICON				FULL LEXICON		
	Model	All	Ambiguous	Rare	Unseen	All	Ambiguous	Rare
Test Data	Das et al. (2014) supervised	82.97	69.27	80.97	23.08			
	Das et al. (2014) best	83.60	69.19	82.31	42.67			
	LOG-LINEAR WORDS	84.53	70.55	81.65	27.27	87.33	70.55	87.19
	LOG-LINEAR EMBEDDING	83.94	70.26	81.03	27.97	86.94	70.26	86.56
	WSABIE EMBEDDING (§3)	**86.49**	**73.39**	**85.22**	**46.15**	**88.41**	**73.10**	**88.93**

Table 2: Frame identification results for FrameNet. See §5.6.

		SEMAFOR LEXICON			FULL LEXICON		
	Model	Precision	Recall	F_1	Precision	Recall	F_1
Development Data	LOG-LINEAR WORDS	76.97	63.37	69.51	77.02	63.55	69.64
	WSABIE EMBEDDING (§3)	**78.33**	**64.51**	**70.75**	**78.33**	**64.53**	**70.76**
Test Data	Das et al. supervised	67.81	60.68	64.05			
	Das et al. best	68.33	61.14	64.54			
	LOG-LINEAR WORDS	71.21	63.37	67.06	73.31	65.20	69.01
	WSABIE EMBEDDING (§3)	**73.00**	**64.87**	**68.69**	**74.29**	**66.02**	**69.91**

Table 3: Full structure prediction results for FrameNet; this reports frame and argument identification performance jointly. We skip LOG-LINEAR EMBEDDING because it underperforms all other models by a large margin.

We search for the stochastic gradient learning rate in $\{\underline{0.0001}, 0.001, 0.01\}$, the margin $\gamma \in \{0.001, \underline{0.01}, 0.1, 1\}$ and the dimensionality of the final vector space $m \in \{256, \underline{512}\}$, to maximize the frame identification accuracy of *ambiguous* lexical units; by ambiguous, we imply lexical units that appear in the training data or the lexicon with more than one semantic frame. The underlined values are the chosen hyperparameters used to analyze the test data.

Argument Candidates The candidate argument extraction method used for the FrameNet data, (as mentioned in §4) was adapted from the algorithm of Xue and Palmer (2004) applied to dependency trees. Since the original algorithm was designed for verbs, we added a few extra rules to handle non-verbal predicates: we added 1) the predicate itself as a candidate argument, 2) the span ranging from the sentence position to the right of the predicate to the rightmost index of the subtree headed by the predicate's head; this helped capture cases like "a *few* months" (where *few* is the predicate and months is the argument), and 3) the span ranging from the leftmost index of the subtree headed by the predicate's head to the position immediately before the predicate, for cases like "your gift *to* Goodwill" (where *to* is the predicate and your gift is the argument).[10]

Frame Lexicon In our experimental setup, we scanned the XML files in the "frames" directory of the FrameNet 1.5 release, which lists all the frames, the corresponding roles and the associated lexical units, and created a frame lexicon to be used in our frame and argument identification models. We noted that this renders every lexical unit as *seen*; in other words, at frame disambiguation time on our test set, for all instances, we only had to score the frames in F_ℓ for a predicate with lexical unit ℓ (see §3 and §5.2). We call this setup FULL LEXICON. While comparing with prior state of the art on the same corpus, we noted that Das et al. (2014) found several unseen predicates at test time.[11] For fair comparison, we took the lexical units for the predicates that Das et al. considered as seen, and constructed a lexicon with only those; training instances, if any, for the unseen predicates under Das et al.'s setup were thrown out as well. We call this setup SEMAFOR LEXICON.[12] We also experimented on the set of unseen instances used by Das et al.

ILP constraints For FrameNet, we used three ILP constraints during argument identification (§4). 1) each span could have only one role, 2) each core role could be present only once, and 3) all overt arguments had to be non-overlapping.

[10]Note that Das et al. (2014) describe the state of the art in FrameNet-based analysis, but their argument identification strategy considered all possible dependency subtrees in a parse, resulting in a much larger search space.

[11]Instead of using the frame files, Das et al. built a frame lexicon from FrameNet's exemplars and the training corpus.

[12]We got Das et al.'s seen predicates from the authors.

Model	All	Ambiguous	Rare
LOG-LINEAR WORDS	94.21	90.54	93.33
LOG-LINEAR EMBEDDING	93.81	89.86	**93.73**
WSABIE EMBEDDING (§3)	**94.79**	**91.52**	92.55
Dev data ↑ ↓ Test data			
Model	All	Ambiguous	Rare
LOG-LINEAR WORDS	**94.74**	**92.07**	**91.32**
LOG-LINEAR EMBEDDING	94.04	90.95	90.97
WSABIE EMBEDDING (§3)	94.56	91.82	90.62

Table 4: Frame identification accuracy results for PropBank. The model and the column names have the same semantics as Table 2.

Model	P	R	F_1
LOG-LINEAR WORDS	80.02	75.58	77.74
WSABIE EMBEDDING (§3)	**80.06**	**75.74**	**77.84**
Dev data ↑ ↓ Test data			
Model	P	R	F_1
LOG-LINEAR WORDS	**81.55**	**77.83**	**79.65**
WSABIE EMBEDDING (§3)	81.32	77.97	79.61

Table 5: Full frame-structure prediction results for Propbank. This is a metric that takes into account frames and arguments together. See §5.7 for more details.

5.5 Experimental Setup for PropBank

Hyperparameters As in §5.4, we made a hyperparameter sweep in the same space. The chosen learning rate was 0.01, while the other values were $\gamma = 0.01$ and $m = 512$. Ambiguous lexical units were used for this selection process.

Argument Candidates For PropBank we use the algorithm of Xue and Palmer (2004) applied to dependency trees.

Frame Lexicon For the PropBank experiments we scanned the frame files for propositions in Ontonotes 4.0, and stored possible core roles for each verb frame. The lexical units were simply the verb associating with the verb frames. There were no unseen verbs at test time.

ILP constraints We used the constraints of Punyakanok et al. (2008).

5.6 FrameNet Results

Table 2 presents accuracy results on frame identification.[13] We present results on all predicates, ambiguous predicates seen in the lexicon or the training data, and rare ambiguous predicates that appear ≤ 11 times in the training data. The WSABIE EMBEDDING model from §3 performs significantly better than the LOG-LINEAR WORDS baseline, while LOG-LINEAR EMBEDDING underperforms in every metric. For the SEMAFOR LEXICON setup, we also compare with the state of the art from Das et al. (2014), who used a semi-supervised learning method to improve upon a supervised latent-variable log-linear model. For unseen predicates from the Das et al. system, we perform better as well. Finally, for the FULL LEXICON setting, the absolute accuracy numbers are even better for our best model. Table 3 presents results on the full frame-semantic parsing task (measured by a reimplementation of the SemEval 2007 shared task evaluation script) when our argument identification model (§4) is used after frame identification. We notice similar trends as in Table 2, and our results outperform the previously published best results, setting a new state of the art.

[13] We do not report partial frame accuracy that has been reported by prior work.

Model	P	R	F_1
LOG-LINEAR WORDS	**77.29**	**71.50**	**74.28**
WSABIE EMBEDDING (§3)	77.13	71.32	74.11
Dev data ↑ ↓ Test data			
Model	P	R	F_1
LOG-LINEAR WORDS	**79.47**	**75.11**	**77.23**
WSABIE EMBEDDING (§3)	79.36	75.04	77.14
Punyakanok et al. *Collins*	75.92	71.45	73.62
Punyakanok et al. *Charniak*	77.09	75.51	76.29
Punyakanok et al. *Combined*	80.53	76.94	78.69

Table 6: Argument only evaluation (semantic role labeling metrics) using the CoNLL 2005 shared task evaluation script (Carreras and Màrquez, 2005). Results from Punyakanok et al. (2008) are taken from Table 11 of that paper.

5.7 PropBank Results

Table 4 shows frame identification results on the PropBank data. On the development set, our best model performs with the highest accuracy on all and ambiguous predicates, but performs worse on rare ambiguous predicates. On the test set, the LOG-LINEAR WORDS baseline performs best by a very narrow margin. See §6 for a discussion.

Table 5 presents results where we measure precision, recall and F_1 for frames and arguments together; this strict metric penalizes arguments for mismatched frames, like in Table 3. We see the same trend as in Table 4. Finally, Table 6 presents SRL results that measures argument performance only, irrespective of the frame; we use the evaluation script from CoNLL 2005 (Carreras and Màrquez, 2005). We note that with a better frame identification model, our performance on SRL improves in general. Here, too, the embedding model barely misses the performance of the best baseline, but we are at par and sometimes better than the single parser setting of a state-of-the-art SRL system (Punyakanok et al., 2008).[14]

[14] The last row of Table 6 refers to a system which used the

6 Discussion

For FrameNet, the WSABIE EMBEDDING model we propose strongly outperforms the baselines on all metrics, and sets a new state of the art. We believe that the WSABIE EMBEDDING model performs better than the LOG-LINEAR EMBEDDING baseline (that uses the same input representation) because the former setting allows examples with different labels and confusion sets to share information; this is due to the fact that all labels live in the same label space, and a single projection matrix is shared across the examples to map the input features to this space. Consequently, the WSABIE EMBEDDING model can share more information between different examples in the training data than the LOG-LINEAR EMBEDDING model. Since the LOG-LINEAR WORDS model always performs better than the LOG-LINEAR EMBEDDING model, we conclude that the primary benefit does not come from the input embedding representation.[15]

On the PropBank data, we see that the LOG-LINEAR WORDS baseline has roughly the same performance as our model on most metrics: slightly better on the test data and slightly worse on the development data. This can be partially explained with the significantly larger training set size for PropBank, making features based on words more useful. Another important distinction between PropBank and FrameNet is that the latter shares frames between multiple lexical units. The effect of this is clearly observable from the "Rare" column in Table 4. WSABIE EMBEDDING performs poorly in this setting while LOG-LINEAR EMBEDDING performs well. Part of the explanation has to do with the specifics of WSABIE training. Recall that the WSABIE EMBEDDING model needs to estimate the label location in $\mathbb{R}^m$ for each frame. In other words, it must estimate 512 parameters based on at most 10 training examples. However, since the input representation is shared across all frames, every other training example from all the lexical units affects the optimal estimate, since they all modify the joint parameter matrix M. By contrast, in the log-linear models each label has its own set of parameters, and they interact only via the normalization constant. The LOG-LINEAR WORDS model does not have this entanglement, but cannot share information between words. For PropBank, these drawbacks and benefits balance out and we see similar performance for LOG-LINEAR WORDS and LOG-LINEAR EMBEDDING. For FrameNet, estimating the label embedding is not as much of a problem because even if a lexical unit is rare, the potential frames can be frequent. For example, we might have seen the SENDING frame many times, even though *telex*.V is a rare lexical unit.

In comparison to prior work on FrameNet, even our baseline models outperform the previous state of the art. A particularly interesting comparison is between our LOG-LINEAR WORDS baseline and the supervised model of Das et al. (2014). They also use a log-linear model, but they incorporate a latent variable that uses WordNet (Fellbaum, 1998) to get lexical-semantic relationships and smooths over frames for ambiguous lexical units. It is possible that this reduces the model's power and causes it to over-generalize. Another difference is that when training the log-linear model, they normalize over all frames, while we normalize over the allowed frames for the current lexical unit. This would tend to encourage their model to expend more of its modeling power to rule out possibilities that will be pruned out at test time.

7 Conclusion

We have presented a simple model that outperforms the prior state of the art on FrameNet-style frame-semantic parsing, and performs at par with one of the previous-best single-parser systems on PropBank SRL. Unlike Das et al. (2014), our model does not rely on heuristics to construct a similarity graph and leverage WordNet; hence, in principle it is generalizable to varying domains, and to other languages. Finally, we presented results on PropBank-style semantic role labeling with a system that included the task of automatic verb frame identification, in tune with the FrameNet literature; we believe that such a system produces more interpretable output, both from the perspective of human understanding as well as downstream applications, than pipelines that are oblivious to the verb frame, only focusing on argument analysis.

Acknowledgments

We thank Emily Pitler for comments on an early draft, and the anonymous reviewers for their valuable feedback.

combination of two syntactic parsers as input.

[15] One could imagine training a WSABIE model with word features, but we did not perform this experiment.

References

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The berkeley framenet project. In *Proceedings of COLING-ACL*.

C. Baker, M. Ellsworth, and K. Erk. 2007. SemEval-2007 Task 19: Frame semantic structure extraction. In *Proceedings of SemEval*.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *Proceedings of CoNLL*.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

D. Das, N. Schneider, D. Chen, and N. A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proceedings of NAACL-HLT*.

D. Das, D. Chen, A. F. T. Martins, N. Schneider, and N. A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

M.-C. de Marneffe and C. D. Manning, 2013. *Stanford typed dependencies manual*.

C. Fellbaum, editor. 1998. *WordNet: an electronic lexical database*.

C. J. Fillmore, C. R. Johnson, and M. R. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

C. J. Fillmore. 1982. Frame Semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: The 90 In *Proceedings of NAACL-HLT*.

R. Johansson and P. Nugues. 2007. LTH: semantic structure extraction using nonprojective dependency trees. In *Proceedings of SemEval*.

A. Klementiev, I. Titov, and B. Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503 – 528.

L. Màrquez, X. Carreras, K. C. Litkowski, and S. Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational Linguistics*, 34(2):145–159.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proceedings of NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-HLT*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, Stroudsburg, PA, USA.

N. Usunier, D. Buffoni, and P. Gallinari. 2009. Ranking with ordered weighted pairwise classification. In *ICML*.

J. Uszkoreit and T. Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-HLT*.

K. Q. Weinberger and L. K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244.

J. Weston, S. Bengio, and N. Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP 2004*.

Y. Zhang and J. Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL-HLT*.

Number	Filename
dev-1	LUCorpus-v0.3__20000420_xin_eng-NEW.xml
dev-2	NTI__SouthAfrica_Introduction.xml
dev-3	LUCorpus-v0.3__CNN_AARONBROWN_ENG_20051101_215800.partial-NEW.xml
dev-4	LUCorpus-v0.3__AFGP-2002-600045-Trans.xml
dev-5	PropBank__TicketSplitting.xml
dev-6	Miscellaneous__Hijack.xml
dev-7	LUCorpus-v0.3__artb_004_A1_E1_NEW.xml
dev-8	NTI__WMDNews_042106.xml
dev-9	C-4__C-4Text.xml
dev-10	ANC__EntrepreneurAsMadonna.xml
dev-11	NTI__LibyaCountry1.xml
dev-12	NTI__NorthKorea_NuclearOverview.xml
dev-13	LUCorpus-v0.3__20000424_nyt-NEW.xml
dev-14	NTI__WMDNews_062606.xml
dev-15	ANC__110CYL070.xml
dev-16	LUCorpus-v0.3__CNN_ENG_20030614_173123.4-NEW-1.xml

Table 7: List of files used as development set for the FrameNet 1.5 corpus.

A Development Data

Table 7 features a list of the 16 randomly selected documents from the FrameNet 1.5 corpus, which we used for development. The resultant development set consists of roughly 4,500 predicates. We use the same test set as in Das et al. (2014), containing 23 documents and 4,458 predicates.