Word Representations, Tree Models and Syntactic Functions

Simon Šuster
University of Groningen
Netherlands
s.suster@rug.nl

Gertjan van Noord
University of Groningen
Netherlands
g.j.m.van.noord@rug.nl

Ivan Titov
University of Amsterdam
Netherlands
titov@uva.nl

arXiv:1508.07709v1 [cs.CL] 31 Aug 2015

Abstract

Word representations induced from models with discrete latent variables (e.g. HMMs) have been shown to be beneficial in many NLP applications. In this work, we exploit labeled syntactic dependency trees and formalize the induction problem as unsupervised learning of tree-structured hidden Markov models. Syntactic functions are used as additional observed variables in the model, influencing both transition and emission components. Such syntactic information can potentially lead to capturing more fine-grain and functional distinctions between words, which, in turn, may be desirable in many NLP applications. We evaluate the word representations on two tasks – named entity recognition and semantic frame identification. We observe improvements from exploiting syntactic function information in both cases, and the results rivaling those of state-of-the-art representation learning methods. Additionally, we revisit the relationship between sequential and unlabeled-tree models and find that the advantage of the latter is not self-evident.

1 Introduction

Word representations have proven to be an indispensable source of features in many NLP systems as they allow better generalization to unseen lexical cases (Koo et al., 2008; Turian et al., 2010; Titov and Klementiev, 2012; Passos et al., 2014; Belinkov et al., 2014). Roughly speaking, word representations allow us to capture semantically or otherwise similar lexical items, be it categorically (e.g. cluster ids) or in a vectorial way (e.g. word embeddings). Although the methods for obtaining word representations are diverse, they normally share the well-known distributional hypothesis (Harris, 1954), according to which the similarity is established based on occurrence in similar contexts. However, word representation methods frequently differ in how they operationalize the definition of context.

Recently, it has been shown that representations using syntactic contexts can be superior to those learned from linear sequences in downstream tasks such as named entity recognition (Grave et al., 2013), dependency parsing (Bansal et al., 2014; Sagae and Gordon, 2009) and PP-attachment disambiguation (Belinkov et al., 2014). They have also been shown to perform well on datasets for intrinsic evaluation, and to capture a different type of semantic similarity than sequence-based representations (Levy and Goldberg, 2014; Šuster and van Noord, 2014; Padó and Lapata, 2007).

Unlike the recent research in word representation learning, focused heavily on word embeddings from the neural network tradition (Collobert and Weston, 2008; Mikolov et al., 2013a; Pennington et al., 2014), our work falls into the framework of hidden Markov models (HMMs), drawing on the work of Grave et al. (2013) and Huang et al. (2014). An attractive property of HMMs is their ability to provide context-sensitive representations, so the same word in two different sentential contexts can be given distinct representations. In this way, we account for various senses of a word.[1] However, this ability requires inference, which is expensive compared to a simple look-up, so we explore in our experiments word representations that are originally obtained in a context-sensitive way, but are then available for look-up as static representations.

Our method includes two types of observed variables: words and syntactic functions. This allows us to address a drawback of learning word representation from unlabeled dependency trees in the

[1] The handling of polysemy and homonymy typically requires extending a model in other frameworks, cf. Huang et al. (2012), Tian et al. (2014) and Neelakantan et al. (2014).

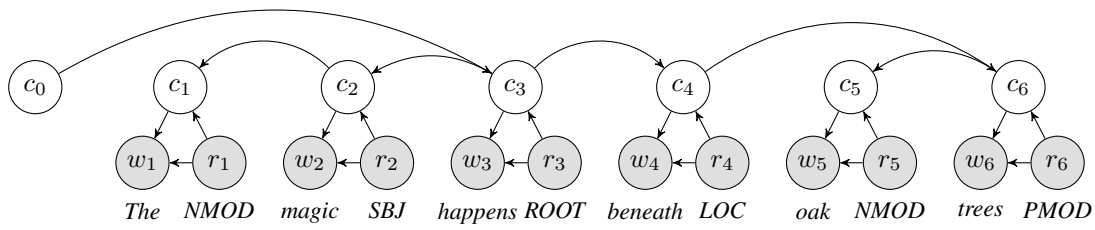

Figure 1: Hidden Markov tree model with syntactic functions, r, as additional observed layer.

context of HMMs (§ 2). The motivation for including syntactic functions comes from the intuition that they act as proxies for semantic roles. The current research practice is to either discard this type of information (so context words are determined on the syntactic structure alone (Grave et al., 2013)), or include it in a preprocessing step, i.e. by attaching syntactic labels to words, as in Levy and Goldberg (2014).

We evaluate the word representations in two structured prediction tasks, named entity recognition (NER) and semantic frame identification. As our extension builds upon sequential and unlabeled-tree HMMs, we also revisit the basic difference between the two, but are unable to entirely corroborate the alleged advantage of syntactic context for word representations in the NER task.

2 Why syntactic functions

A word can typically occur in distinct syntactic functions. Since these account for words in different semantic roles (Bender, 2013; Levin, 1993), the incorporation of the syntactic function between the word and its parent could give us more precise representations. For example, in "Carla bought the computer", the subject and the object represent two different semantic roles, namely the buyer and the goods, respectively. Along similar lines, Padó and Lapata (2007), Šuster and van Noord (2014) and Grave et al. (2013) argue that it is inaccurate to treat all context words as equal contributors to a word's meaning.

In HMM learning, the parameters obtained from training on unlabeled syntactic structure encode the probabilistic relationship between the hidden states of parent and child, and that between the hidden state and the word. The tree structure thus only defines the word's context, but is oblivious of the relationship between the words. For example, Grave et al. (2013) acknowledge precisely this limitation of their unlabeled-tree representations by providing as example the hidden state of a verb, which cannot discriminate between left (e.g. subject) and right (e.g. object) neighbors because of shared transition parameters. This adversely affects the accuracy of their super-sense tagger for English. Similarly, Šuster and van Noord (2014) show that filtering dependency instances based on syntactic functions can positively affect the quality of obtained Brown word clusters when measured in a wordnet similarity task.

3 A tree model with syntactic functions

We represent a sentence as a tuple of K words, $\mathbf{w} = (w_1, \ldots, w_K)$, where each $w_k \in \{1, \ldots, |\mathcal{V}|\}$ is an integer representing a word in the vocabulary $\mathcal{V}$. The goal is to infer a tuple of K states $\mathbf{c} = (c_1, \ldots, c_K)$, where each $c_k \in \{1, \ldots, N\}$ is an integer representing a semantic class of w_k, and N is the number of states, which needs to be set prior to training. Another possibility is to let w_k's representation be a probability distribution over N states. In this case, we denote w_k's representation as $\mathbf{u}_k \in \mathbb{R}^N$.

As usual in Markovian models, the generation of the sentence can be decomposed into the generation of classes (transitions) and the generation of words (emissions). The process is defined on a tree, in which a node c_k is generated by its single parent $c_{\pi(k)}$, where $\pi : \{1, \ldots, K\} \mapsto \{0, \ldots, K\}$, with 0 representing the root of the tree (the only node not emitting a word). We denote a syntactic function as $r \in \{r_1, \ldots, r_S\}$, where S is the total number of syntactic function types produced by the syntactic parser. We encode the syntactic function at position k as $r_k \triangleq r_{w_k \to w_{\pi(k)}}$, i.e. the dependency relation between w_k and its parent.

We would like the variable r to modulate the transition and emission processes. We achieve this by drawing on the Input-output HMM architecture of Bengio and Frasconi (1996), who introduce a sequential model in which an additional sequence

of observations called input becomes part of the model, and the model is used as a conditional predictor. The authors describe the application of their model in speech processing, where the goal is to obtain an accurate predictor of the output phoneme layer from the input acoustic layer. Our focus is, in contrast, on representation learning (hidden layer) rather than prediction (output layer). Also, we adapt their sequential topology to trees.

The probability distribution of words and semantic classes is conditional on syntactic functions and is factorized as:

$$p(\mathbf{w}, \mathbf{c}|\mathbf{r}) = \prod_{k=1}^{K} p(w_k|c_k, r_k) p(c_k|c_{\pi(k)}, r_k), \tag{3.1}$$

where r_k encodes additional information about w_k, in our case the syntactic function of w_k to its parent. This is represented graphically in fig. 1.

The parameters of the model are stored in column-stochastic transition and emission matrices[2]:

$\mathbf{T}$, where $T_{ijl} = p(c_k{=}i \mid c_{\pi(k)}{=}j, r_k{=}l)$
$\mathbf{O}$, where $O_{ijl} = p(w_k{=}i \mid c_k{=}j, r_k{=}l)$

The number of required parameters for representing the transitions is $O(N^2\, S)$, and for representing the emissions $O(N\, |\mathcal{V}|\, S)$.

Our model satisfies the single-parent constraint and can be applied to proper trees only. It is in principle possible to extend the base representation for the model by using approximate inference techniques that work on graphs (Murphy, 2012, p. 720), but we do not explore this possibility here.[3]

As opposed to an unlabeled-tree HMM, our extension can in fact be categorized as an *in*homogeneous model since the transition and emission probability distributions change as a function of input, cf. Bengio (1999). Another comparison concerns the learning of long-term dependencies: since in the Input-output architecture the transition probabilities can change as a function of input at each k, they can be more deterministic (have lower entropy) than the transition probabilities of an HMM. Having the transition parameters closer to zero or one reduces the ambiguity of the next state and allows the context to flow more easily. A concrete graphical example is given in fig. 2.

[2]We are abusing the terminology slightly, as these are in fact three-dimensional arrays.

[3]This would be relevant for dependency annotation schemes which include secondary edges.

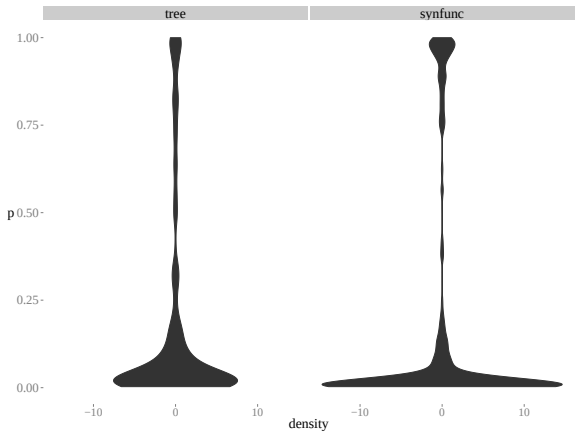

Figure 2: The transition probabilities of a tree HMM with syntactic functions (*synfunc*) are sparser and have a lower entropy (5.34) than those of an unlabeled-tree HMM (*tree*; entropy of 5.6).

4 Learning and inference

We train the model with the Expectation-Maximization (EM) algorithm (Baum, 1972) and use the sum-product message passing for inference on trees (Pearl, 1988). The inference procedure (the estimation of hidden states) is the same as in an unlabeled-tree model, except that it is performed conditionally on r.

The parameters $\mathbf{T}$ and $\mathbf{O}$ are estimated with maximum likelihood estimation. In the E-phase, we obtain pseudo-counts from the existing parameters, as shown in fig. 3. The M-step then normalizes the transition pseudo-counts (and similarly for emissions):

$$\hat{T}_{ijl} = \frac{\tau_{ijl}}{\sum_j \tau_{ij'l}} \tag{4.1}$$

4.1 State splitting and merging

We explore the idea of introducing complexity gradually in order to alleviate the problem of EM finding a poor solution, which can be particularly severe when the search space is large (Petrov et al., 2006). The splitting procedure starts with a small number of states, splits the parameters of each state s into s_1 and s_2 by cloning s and slightly perturbing. The model is retrained, and a new split round takes place. To allow splitting states to various degrees, Petrov et al. also merge back those split states which improve the likelihood the least. Although the merge step is done approximately and does not require new cycles of inference, we find that the extra running time does not justify the spo-

$$\tau_{ijl} = \sum_{n=1}^{N} \sum_{k=1}^{K_n} \mathbb{E}\Big[1\{C_k^{(n)}{=}i, C_{\pi(k)}^{(n)}{=}j, R_k^{(n)}{=}l\} \;\Big|\; W^{(n)}{=}\mathbf{w}^{(n)}, R^{(n)}{=}\mathbf{r}^{(n)}\Big]$$

$$\omega_{ijl} = \sum_{n=1}^{N} \sum_{k=1}^{K_n} \mathbb{E}\Big[1\{W_k^{(n)}{=}i, C_{\pi(k)}^{(n)}{=}j, R_k^{(n)}{=}l\} \;\Big|\; W^{(n)}{=}\mathbf{w}^{(n)}, R^{(n)}{=}\mathbf{r}^{(n)}\Big]$$

Figure 3: Obtaining pseudo-counts, or expected sufficient statistics, in the E-step.

radic improvements we observe. We settle therefore on the splitting-only regime.

4.2 Decoding for HMM-based models

Once a model is trained, we can search for the most probable states given observed data by using the max-product message passing (MAX-PRODUCT, a generalization of Viterbi) for efficient decoding on trees: $\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} p(C = \mathbf{c} \mid W = \mathbf{w}, R = \mathbf{r})$.

We have also tried posterior (or minimum risk) decoding (Lember and Koloydenko, 2014; Ganchev et al., 2008), but without consistent improvements.

The search for the best states can be avoided by taking the posterior state distribution $\mathbf{u}_k$ over N hidden states (Nepal and Yates, 2014; Grave et al., 2014): $u_c^{(k)} = \mathbb{E}\big[1\{C_k = c\} \mid W = \mathbf{w}, R = \mathbf{r}\big]$. We call this vectorial representation POST-TOKEN.

In both cases, inference is performed on a concrete sentence, thus providing a context-sensitive representation. We find in our experiments that POST-TOKEN consistently outperforms MAX-PRODUCT due to its ability to carry more information and uncertainty. This can then be exploited by the downstream task predictor.

One disadvantage of obtaining context-sensitive representations is the relatively costly inference. The inference and decoding are also sometimes not applicable, such as in information retrieval, where the entire sentence is usually not given (Huang et al., 2011). A trade-off between full context sensitivity and efficiency can be achieved by considering a static representation (POST-TYPE). It is obtained in a context-insensitive way (Huang et al., 2011; Grave et al., 2014) by averaging posterior state distributions (context-sensitive) of all occurrences of a word type $\tilde{w}$ from a large corpus $\mathcal{U}$:

$$\mathbf{v}^{(\tilde{w})} = \frac{1}{Z_{\tilde{w}}} \sum_{i \in \mathcal{U}: w_i = \tilde{w}} \mathbf{u}^{(i)}. \tag{4.2}$$

In fig. 4, we give a graphical example of the word representations learned with our model (§ 5.5), obtained either with the POST-TOKEN or the POST-TYPE. To visualize the representations, we apply multidimensional scaling.[4] The model clearly separates between management positions and parts of body, and interestingly, puts "head" closer to management positions, which can be explained by the business and economic nature of the Bllip corpus. The words "chief" and "executive" are located together, yet isolated from others, possibly because of their strong tendency to precede nouns. The arrow on the plot indicates the shift in the meaning when a POST-TOKEN representation is obtained for "head" (part-of-body) within a sentence.

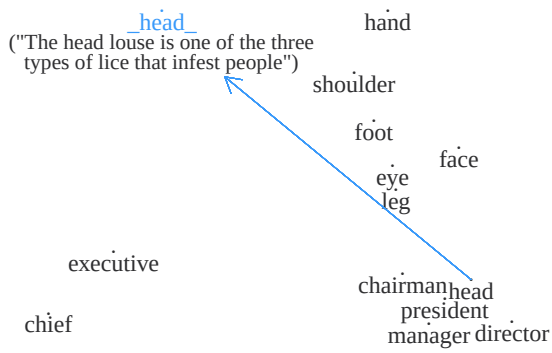

Figure 4: Representations obtained with our model with syntactic functions. All are static POST-TYPE representations, except "_head_", which is obtained with POST-TOKEN from the concrete sentence.

Despite the advantage of POST-TOKEN to account for word senses, we observe that POST-TYPE performs better in almost all experiments. A likely explanation is that averaging increases the generalizability of representations. For the concrete tasks in which we apply the word representations, the increased robustness is simply more important than context sensitivity. Also, POST-TYPE might be less sensitive to parsing errors during test time.

[4] https://github.com/scikit-learn/scikit-learn

5 Empirical study

5.1 Parameters and setup

We observe faster convergence times with online EM, which updates the parameters more frequently. Specifically, we use the mini-batch step-wise EM (Liang and Klein, 2009; Cappé and Moulines, 2009), and determine the hyper-parameters on the held-out dataset of 10,000 sentences to maximize the log-likelihood. We find out that higher values for the step-wise reduction power α and the mini-batch size lead to better overall log-likelihood, but with a somewhat negative effect on the convergence speed. We finally settle on $\alpha = \{0.6, 0.7, 0.8, 0.9, \underline{1}\}$ and mini-batch size of $\{1, 10, 100, \underline{1000}\}$ sentences. We find that a couple of iterations over the entire dataset is sufficient to obtain good parameters, cf. Klein (2005).

Initialization. Since the EM algorithm in our setting only finds a local optimum of the log-likelihood, the initialization of model parameters can have a major impact on the final outcome. We initialize the emission matrices with Brown clusters by first assigning random values between 0 and 1 to the matrix elements, and then multiplying those elements that represent words in a cluster by a factor of $f \in \{10, 100, \underline{1K}, 10K\}$. Finally, we normalize the matrices. This technique incorporates a strong bias towards word-class emissions that exist (deterministically) in Brown clusters. The transition parameters are simply set to random numbers sampled from the uniform distribution between 0 and 1, and finally normalized.

Approximate inference. Following Grave et al. (2013) and Pal et al. (2006), we approximate the belief vectors during inference,[5] which speeds up learning and works as regularization. We use the k-best projection method, in which only k-largest coefficients (in our case $k = \frac{1}{8}N$) are kept.

5.2 Data for obtaining word representations

English. We use the 43M-word Bllip corpus (Charniak et al., 2000) of WSJ texts, from which we remove the sentences of the PTB and those whose length is ≤ 4 or ≥ 40. We use the MST dependency parser (McDonald and Pereira, 2006) for English and build a projective, second order model, trained on sections 2–21 of the Penn Treebank WSJ (PTB). Prior to that, the PTB was patched with NP bracketing rules (Vadas and Curran, 2007) and converted to dependencies with LTH (Johansson and Nugues, 2007). The parser achieves the unlabeled/labeled accuracy of 91.5/85.22 on section 23 of the PTB without retagging the POS. For POS-tagging the Bllip corpus and the evaluation datasets, we use the Citar tagger [6]. After parsing, we replace the words occurring less than 40 times with a special symbol to model OOV words. This results in the vocabulary size of 27,000 words.

Dutch. We first produce a random sample of 2.5M sentences from the SoNaR corpus[7], then follow the same preprocessing steps as for English. We parse the corpus with Alpino (van Noord, 2006), an HPSG parser with a maxent disambiguation component. In contrast with English, for which we use word forms, we keep here the root forms produced by the parser's lexical analyzer. The resulting vocabulary size is 25,000 words. The analyses produced by the parser represent multiple parents to facilitate the treatment of wh-clauses, coordination and passivization. Since our method expects proper trees, we convert the parser output to CoNLL format.[8]

5.3 Evaluation tasks

Named entity recognition. We use the standard CoNLL-2002 shared task dataset for Dutch and CoNLL-2003 dataset for English. We also include the out-of-domain MUC-7 testset, preprocessed according to Turian et al. (2010). We refer the reader to Ratinov and Roth (2009) for a detailed description of the NER classification problem.

Just like Turian et al. (2010), we use the averaged structured perceptron (Collins, 2002) with Viterbi as the base for our NER system.[9] The classifier is trained for a fixed number of iterations, and uses these baseline features:

- w_k information: is-alphanumeric, all-digits, all-capitalized, is-capitalized, is-hyphenated;
- prefixes and suffixes of w_k;
- word window: $w_k, w_{k\pm1}, w_{k\pm2}$;
- capitalization pattern in the word window.

We construct N real-valued features for a word vector of dimensionality N, and a simple indicator feature for a categorical word representation.

[5] In a bottom-up pass, a belief vector represents the local evidence by multiplying the messages received from the children of a node, as well as the emission probability at that node.

[6] http://github.com/danieldk/citar

[7] http://lands.let.ru.nl/projects/SoNaR

[8] http://www.let.rug.nl/bplank/alpino2conll/

[9] http://github.com/LxMLS/lxmls-toolkit

Semantic frame identification. Frame-semantic parsing is the task of identifying *a*) semantic frames of predicates in a sentence (given *target* words evoking frames), and *b*) frame arguments participating in these events (Fillmore, 1982; Das et al., 2014). Compared to NER, in which the classification decisions apply to a relatively small set of words, the problem of semantic frame identification concerns making predictions for a broader set of words (verbs, nouns, adjectives, sometimes prepositions).

Figure 5: A parse with HINDERING and CAUSE_TO_MAKE_PROGRESS frames with respective arguments.

We use the Semafor parser (Das et al., 2014) consisting of two log-linear components trained with gradient-based techniques. The parser is trained and tested on the FrameNet 1.5 full-text annotations. Our test set consists of the same 23 documents as in Hermann et al. (2014). We investigate the effect of word representation features on the frame identification component. We measure Semafor's performance on gold-standard targets, and report the accuracy on *exact matches*, as well as on *partial matches*. The latter give partial credit to identified related frames. We use and modify the publicly available implementation at `http://github.com/sammthomson/semafor`.

Our baseline features for a target w_k include:

- w_k and $w_{\pi(k)}$ (if the parent is a preposition, the grandparent is taken by collapsing the dependency),
- their lemmas and POS tags,
- syntactic functions between:
  - w_k and its children,
  - w_k and $w_{\pi(k)}$ (added by ourselves),
  - $w_{\pi(k)}$ and its parent $w_{\pi(\pi(k))}$.

5.4 Baseline word representations

We test our model, which we call SYNFUNC-HMM, against the following baselines:

- BASELINE: no word representation features
- HMM: a sequential HMM
- TREE-HMM: a tree HMM

We induce other representations for comparison:

- BROWN: Brown clusters
- DEP-BROWN: dependency Brown clusters
- SKIP-GRAM: Skip-Gram word embeddings

5.5 Preparing word representations

Brown clusters. Brown clusters (Brown et al., 1992) are known to be effective and robust when compared, for example, to word embeddings (Bansal et al., 2014; Passos et al., 2014; Nepal and Yates, 2014; Qu et al., 2015). The method can be seen as a special case of a HMM in which word emissions are deterministic, i.e. a word belongs to at most one semantic class. Recently, an extension has been proposed on the basis of a dependency language model (Šuster and van Noord, 2014). We use the publicly available implementations.[10]

Following other work on English (Koo et al., 2008; Nepal and Yates, 2014), we add both coarse- and fine-grained clusters as features by using prefixes of length 4, 6, 10 and 20 in addition to the complete binary tree path. For Dutch, coarser-grained clusters do not yield any improvement. Brown features are included in a window around the target word, just as the NER word features. When adding cluster features to the frame-semantic parser, we transform the cluster identifiers to one-hot vectors, which gives a small improvement over the use of indicator features.

HMM-based models. The N-dimensional representations obtained from HMMs and their variants are included as N distinct continuous features. In the NER task, word representations are included at w_k and w_{k+1} for Dutch and at w_k for English, which we determined on the development set. We investigate state space sizes of 64, 128 and 256 and finally choose N=128 as a reasonable trade-off between training time and quality. We use the same dimensionality for other word representation models in this paper.

We observe that by constraining SYNFUNC-HMM to use only the k most frequent syntactic functions and to treat the remaining ones as a single special syntactic function, we obtain better results in our evaluation tasks. This is because for a model with all S syntactic functions produced by the parser, there is less learning evidence for more infrequent syntactic functions. We explore the effect of keeping up to five most frequent syntactic functions, ignoring functional ones such as

[10] `http://github.com/percyliang/brown-cluster`, `http://github.com/rug-compling/dep-brown-cluster`

	English			Dutch			MUC test set		
Model	P	R	F-1	P	R	F-1	F-1	F-1_{type}	F-1_{unlab}
BASELINE	80.12	77.30	78.69	75.36	70.92	73.07	65.44	87.04	96.25
HMM	**81.49**	78.90	**80.17**	77.61	73.97	75.74	70.20	87.66	96.50
TREE-HMM	80.49	78.10	79.28	77.41	73.48	75.40	65.67	86.99	96.53
SYNFUNC-HMM	80.65	78.90	79.76 (+.48)	**78.54**	**75.23**	**76.85** (+1.45)	66.49 (+.82)	86.93 (-.06)	**96.69** (+.16)
BROWN	80.15	77.28	78.70	77.88	71.73	74.68	68.85	87.72	96.67
DEP-BROWN	78.80	75.73	77.23	77.50	73.66	75.53	68.31	87.44	96.47
SKIP-GRAM	80.80	**78.98**	79.88	76.02	71.28	73.57	**72.42**	**88.94**	**96.69**

Table 1: NER results (precision, recall and F-score) on English and Dutch test sets. Best result per column in bold. The score increase reported in parentheses is in comparison to TREE-HMM. F-1_{type} is the F-score measured per word type, and F-1_{unlab} is the F-score measured per word type, ignoring labels.

punctuation and determiner.[11] The final selection is shown in table 2.

English	Dutch
nmod (nominal modifier)	mod (modifier)
pmod (prepositional modifier)	su (subject)
sub (subject)	obj1 (direct object)
	cnj (conjunction)
	mwp (multiword unit)

Table 2: Syntactic functions in SYNFUNC-HMM for English (produced by the MST parser) and Dutch (produced by Alpino).

For NER experiments, the representations from all HMM models are obtained with three different "decoding" methods (§ 4.2). Since POST-TYPE is performing best overall, we only report the results for this method in the evaluation.[12]

Word embeddings. We use the Skip-Gram model presented in Mikolov et al. (2013a) (`https://code.google.com/p/word2vec/`), trained with negative sampling (Mikolov et al., 2013b). The training seeks to maximize the dot product between word-context pairs encountered in the training corpus and minimize the dot product between those pairs in which the context word is randomly sampled. We set both the number of negative examples and the size of the context window to 5, the downsampling threshold to 1×10^{-4}, and the number of iterations to 15.

5.6 NER results

The results for all testsets are shown in table 1. For English, all HMM-based models improve the baseline, with the sequential HMM achieving the highest F-score. Our SYNFUNC-HMM performs on a par with SKIP-GRAM. It outperforms the unlabeled-tree model, indicating that the added observations are useful and correctly incorporated. Brown clusters do not exceed the BASELINE score.[13] Testing for significance with a bootstrap method (Søgaard et al., 2014), we find out that only HMM improves significantly at $p < 0.01$ on macro-F1 over BASELINE, while SKIPGRAM and SYNFUNC-HMM show significant improvements only for the location entity type.

The general trend for Dutch is somewhat different. Most notably, all word representations contribute much more effectively to the overall classification performance compared to English. The best-scoring model, our SYNFUNC-HMM, improves over the baseline significantly by as much as about 3.8 points. Part of the reason SYNFUNC-HMM works so well in this case is that it can make use of the informative "mwp" syntactic function between the parts of a multiword unit. Similarly as for English, the unlabeled-tree HMM performs slightly worse than the sequential HMM. The cluster features are more valuable here than in English, and we also observe a 0.7-point advantage by using dependency Brown clusters over the standard, bigram Brown clusters. The SKIP-GRAM model does not perform as well as in English, which might indicate that the hyper-parameters would need fine-tuning specific to Dutch.

On the out-of-domain MUC dataset, tree-based representations appear to perform poorly, whereas the highest score is achieved by the SKIP-GRAM method. Unfortunately, it is difficult to generalize from these F-1 results alone. Concretely,

[11] We define the list of function-marker syntactic functions following Goldberg and Orwant (2013).

[12] While exploring the constraint on the number of syntactic functions, we do find that POST-TOKEN outperforms POST-TYPE in some sets of syntactic functions, but not in the final, best-performing selection.

[13] However, after additional experiments we observe that the cluster features do improve over the baseline score when the number of clusters is increased.

the dataset contains 3,518 named entities, and the SKIP-GRAM method makes 258 correct predictions more than TREE-HMM. However, because the MUC dataset covers the narrow topic of missile-launch scenarios, the system gets badly penalized if a mistake is made repeatedly for a certain named entity. For example, only the entity "NASA" occurs 103 times, most of which are wrongly classified by the TREE-HMM system, but correctly by SKIP-GRAM. The overall performance may therefore hinge on a limited number of frequently occurring entities. A workaround is to evaluate per *entity type* — calculate the F-score for each entity, then average over all entity types. The results for this evaluation scenario are reported as F-1_{type}. SKIP-GRAM still performs best, but the difference to other models is smaller. Finally, we also report F-1_{unlab}, calculated as F-1_{type} but ignoring the actual entity label. So, if a named-entity token is recognized as such, we count it as correct prediction ignoring the entity label type, similarly as done by Ratinov and Roth (2009). Since SYNFUNC-HMM performs better here, we can conclude that it is more effective at identifying entities rather than at labeling them.

The fact that we observe different tendencies for English and Dutch can be attributed to an interplay of factors, such as language differences (Bender, 2011), differently-performing syntactic parsers, and differences specific to the evaluation datasets. We briefly discuss the first possibility. It is clear from table 1 that all syntax-based models (DEP-BROWN, TREE-HMM, SYNFUNC-HMM) generally benefit Dutch more than English. We hypothesize that since the word order in Dutch is generally less fixed than in English,[14] a sequence-based model for Dutch cannot capture selectional preferences that successfully, i.e. there is more interchanging of semantically diverse words in a small word window. This then makes the difference in performance between sequential and tree models more apparent for Dutch.

5.7 Semantic frame identification results

The results are shown in table 3. The best score is obtained by the Skip-Gram embeddings, however, the difference to other models outperforming the baseline is small. For example, SKIP-GRAM correctly identifies only two cases more than DEP-BROWN, out of 3681 correctly disambiguated frames.

The SYNFUNC-HMM model outperforms all other HMM models in this task. The differences are larger when scoring partial matches.[15]

Model	Exact	Partial
BASELINE	82.70	90.44
HMM	82.20	90.20
TREE-HMM	82.89	90.59
SYNFUNC-HMM	82.95 (+0.06)	90.80 (+0.21)
BROWN	83.15	90.74
DEP-BROWN	83.15	90.76
SKIP-GRAM	**83.19**	**90.91**

Table 3: Frame identification accuracy. Score increase in parentheses is relative to TREE-HMM.

5.8 Further discussion

We can conclude from the NER experiments that *unlabeled* syntactic trees do not in general provide a better structure for defining the contexts compared to plain sequences. The only exception is the case of dependency Brown clustering for Dutch. Comparing our results to those of Grave et al. (2013), we therefore cannot confirm the same advantage when using unlabeled-tree representations. In semantic frame identification, however, the unlabeled-tree representations do compare more favorably to sequential representations.

Our extension with syntactic functions outperforms the baseline and other HMM-based representations in practically all experiments. It also outperforms all other word representations in Dutch NER. The advantage comes from discriminating between the types of contexts, for example between a modifier and a subject, which is impossible in sequential or unlabeled-tree HMM architectures. The results for English are comparable to those of the state-of-the-art representation methods.

6 Related work

HMMs have been used successfully for learning word representations already before, see Huang et al. (2014) for an overview, with an emphasis on investigating domain adaptability. Models with more

[14] For instance, it is unusual for the direct object in English to precede the verb, but quite common in Dutch.

[15] On exact matches, only DEP-BROWN and BROWN significantly outperform the baseline with the $p < 0.05$. On partial matches, DEP-BROWN, BROWN, SKIP-GRAM and SYNFUNC-HMM all outperform the baseline significantly. SYNFUNC-HMM performs significantly better than TREE-HMM on partial matches, whereas the difference between SKIP-GRAM and SYNFUNC-HMM is not significant. The significance tests were run using paired permutation.

complex architecture have been proposed, such as a factorial HMM (Nepal and Yates, 2014), trained using approximate variational inference and applied to POS tagging and chunking. Recently, semantic compositionality of HMM-based representations based on the framework of distributional semantics has been investigated by Grave et al. (2014).

There is a long tradition of unsupervised training of HMMs for POS tagging (Kupiec, 1992; Merialdo, 1994), with more recent work on incorporating bias by favoring sparse posterior distributions within the posterior regularization framework (Graça et al., 2007), and for example on auto-supervised refinement of HMMs (Garrette and Baldridge, 2012). It would be interesting to see how well these techniques could be applied to word representation learning methods like ours.

The extension of HMMs to dependency trees for the purpose of word representation learning was first proposed by Grave et al. (2013). Although our baseline HMM methods, HMM and TREE-HMM, conceptually follow the models of Grave et al., there are still several practical differences. One source of differences is in the precise steps taken when performing Brown initialization, state splitting, and also approximation of belief vectors during inference. Another source involves the evaluation setting. Their NER classifier uses only a single feature, and the inclusion of Brown clusters does not make use of the clustering hierarchy. In this respect, our experimental setting is more similar to Turian et al. (2010). Another practical difference is that Grave et al. concatenate words with POS-tags to construct the input text, whereas we use tokens (English) or word roots (Dutch).

The incorporation of word representations into semantic frame identification has been explored in Hermann et al. (2014). They perform a projection of generic word embeddings for context words to a low-dimensional representation, which also learns an embedding for each frame label. The method selects the frame closest to the low-dimensional representation obtained through mapping of the input embeddings. Their approach differs from ours in that they induce new representations that are tied to a specific application, whereas we aim to obtain linguistically enhanced word representations that can be subsequently used in a variety of tasks. In our case, the word representations are thus included as additional features in the log-linear model. The inclusion accounts for syntactic functions between the target and its context words. Although Hermann et al. also use syntactic functions, they are used to position the general word embeddings within a single input context embedding. Unfortunately, we are unable to directly compare our results with theirs as their parser implementation is proprietary. The accuracy of our baseline system on the test set is 0.27 percent lower in the exact matching regime and 0.07 lower in the partial matching regime compared to the baseline implementation (Das et al., 2014) they used.[16]

The topic of context type (syntactic vs. linear) has been abundantly treated in distributional semantics (Lin, 1998; Baroni and Lenci, 2010; van de Cruys, 2010) and elsewhere (Boyd-Graber and Blei, 2008; Tjong Kim Sang and Hofmann, 2009).

7 Conclusion

We have proposed an extension of a tree HMM with syntactic functions. The obtained word representations achieve better performance than those from the unlabeled-tree model. Our results also show that simply preferring an unlabeled-tree model over a sequential model does not always lead to an improvement. An important direction for future work is to investigate how discriminating between context types can lead to more accurate models in other frameworks.

Acknowledgements

We would like to thank Edouard Grave and Sam Thomson for valuable discussion and suggestions.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL*.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Leonard E. Baum. 1972. An inequality and associated maximization technique in statistical estimation for probalistic functions of Markov processes. In *Inequalities*.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for preposi-

[16] Among other implementation differences, they introduce a variable capturing lexico-semantic relations from WordNet.

tional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572.

Emily M. Bender. 2011. On Achieving and Evaluating Language-Independence in NLP. *Linguistic Issues in Language Technology*, 6(3):1–26.

Emily Bender. 2013. *Linguistic Fundamentals for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Yoshua Bengio and Paolo Frasconi. 1996. Input-output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5).

Yoshua Bengio. 1999. Markovian models for sequential data. *Neural computing surveys*, 2:129–162.

Jordan Boyd-Graber and David M. Blei. 2008. Syntactic topic models. In *NIPS*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Olivier Cappé and Eric Moulines. 2009. Online em algorithm for latent data models. *Journal of the Royal Statistical Society*.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. *BLLIP 1987–1989 WSJ Corpus Release 1, LDC No. LDC2000T43*. Linguistic Data Consortium.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP*.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *ICML*.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Charles Fillmore. 1982. Frame semantics. *Linguistics in the morning calm*, pages 111–137.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *ACL-HLT*.

Dan Garrette and Jason Baldridge. 2012. Type-Supervised Hidden Markov Models for Part-of-Speech Tagging with Incomplete Tag Dictionaries. In *EMNLP*.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In **SEM*.

Edouard Grave, Guillaume Obozinski, and Francis Bach. 2013. Hidden Markov tree models for semantic class induction. In *CoNLL*.

Edouard Grave, Guillaume Obozinski, and Francis Bach. 2014. A Markovian approach to distributional semantics with application to semantic compositionality. In *COLING*.

João Graça, Kuzman Ganchev, and Ben Taskar. 2007. Expectation maximization and posterior constraints. In *NIPS*.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *ACL*.

Fei Huang, Alexander Yates, Arun Ahuja, and Doug Downey. 2011. Language models as representations for weakly-supervised nlp tasks. In *CoNLL*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*.

Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. 2014. Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*, 40(1):85–120.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *NODALIDA*, pages 105–112, Tartu, Estonia.

Dan Klein. 2005. *The unsupervised learning of natural language structure*. Ph.D. thesis, Stanford University.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL-HLT*.

Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3):225–242.

Jüri Lember and Alexey A. Koloydenko. 2014. Bridging Viterbi and posterior decoding: A generalized risk approach to hidden path inference based on Hidden Markov models. *Journal of Machine Learning Research*, 15(1):1–58.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL*.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *HLT-NAACL*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Bernard Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR Workshop Papers*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

Anjan Nepal and Alexander Yates. 2014. Factorial Hidden Markov models for learning representations of natural language. In *ICLR*.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33:161–199.

Chris Pal, Charles Sutton, and Andrew McCallum. 2006. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *ICASSP*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. *EMNLP*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. Big Data Small Data, In Domain Out-of Domain, Known Word Unknown Word: The Impact of Word Representation on Sequence Labelling Tasks. *arXiv preprint arXiv:1504.05319*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Kenji Sagae and Andrew S. Gordon. 2009. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *IWPT*.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. 2014. What's in a p-value in NLP? In *CoNLL*.

Simon Šuster and Gertjan van Noord. 2014. From neighborhood to parenthood: the advantages of dependency representation over bigrams in Brown clustering. In *COLING*.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*.

Ivan Titov and Alexandre Klementiev. 2012. A Bayesian approach to unsupervised semantic role induction. In *EACL*.

Erik Tjong Kim Sang and Katja Hofmann. 2009. Lexical patterns or dependency patterns: Which is better for hypernym extraction? In *CoNLL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*.

David Vadas and James R. Curran. 2007. Adding Noun Phrase Structure to the Penn Treebank. In *ACL*.

Tim van de Cruys. 2010. *Mining for Meaning: The Extraction of Lexico-semantic Knowledge from Text*. Ph.D. thesis, University of Groningen.

Gertjan van Noord. 2006. At Last Parsing Is Now Operational. In *TALN*.