A practical and linguistically-motivated approach to compositional distributional semantics

Denis Paperno and **Nghia The Pham** and **Marco Baroni**
Center for Mind/Brain Sciences (University of Trento, Italy)
(denis.paperno|thenghia.pham|marco.baroni)@unitn.it

Abstract

Distributional semantic methods to approximate word meaning with context vectors have been very successful empirically, and the last years have seen a surge of interest in their compositional extension to phrases and sentences. We present here a new model that, like those of Coecke et al. (2010) and Baroni and Zamparelli (2010), closely mimics the standard Montagovian semantic treatment of composition in distributional terms. However, our approach avoids a number of issues that have prevented the application of the earlier linguistically-motivated models to full-fledged, real-life sentences. We test the model on a variety of empirical tasks, showing that it consistently outperforms a set of competitive rivals.

1 Compositional distributional semantics

The research of the last two decades has established empirically that distributional vectors for words obtained from corpus statistics can be used to represent word meaning in a variety of tasks (Turney and Pantel, 2010). If distributional vectors encode certain aspects of word meaning, it is natural to expect that similar aspects of sentence meaning can also receive vector representations, obtained compositionally from word vectors. Developing a practical model of compositionality is still an open issue, which we address in this paper. One approach is to use simple, parameter-free models that perform operations such as pointwise multiplication or summing (Mitchell and Lapata, 2008). Such models turn out to be surprisingly effective in practice (Blacoe and Lapata, 2012), but they have obvious limitations. For instance, symmetric operations like vector addition are insensitive to syntactic structure, therefore meaning differences encoded in word order are lost in composition: *pandas eat bamboo* is identical to *bamboo eats pandas*. Guevara (2010), Mitchell and Lapata (2010), Socher et al. (2011) and Zanzotto et al. (2010) generalize the simple additive model by applying structure-encoding operators to the vectors of two sister nodes before addition, thus breaking the inherent symmetry of the simple additive model. A related approach (Socher et al., 2012) assumes richer lexical representations where each word is represented with a vector and a matrix that encodes its interaction with its syntactic sister. The training proposed in this model estimates the parameters in a supervised setting. Despite positive empirical evaluation, this approach is hardly practical for general-purpose semantic language processing, since it requires computationally expensive approximate parameter optimization techniques, and it assumes task-specific parameter learning whose results are not meant to generalize across tasks.

1.1 The lexical function model

None of the proposals mentioned above, from simple to elaborate, incorporates in its architecture the intuitive idea (standard in theoretical linguistics) that semantic composition is more than a weighted combination of words. Generally one of the components of a phrase, e.g., an adjective, acts as a function affecting the other component (e.g., a noun). This underlying intuition, adopted from formal semantics of natural language, motivated the creation of the *lexical function* model of composition (*lf*) (Baroni and Zamparelli, 2010; Coecke et al., 2010). The lf model can be seen as a projection of the symbolic Montagovian approach to semantic composition in natural language onto the domain of vector spaces and linear operations on them (Baroni et al., 2013). In lf, arguments are vectors and functions taking arguments (e.g., adjectives that combine with nouns) are tensors, with the number of arguments (n) determining the

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 90–99,
Baltimore, Maryland, USA, June 23-25 2014.

order of tensor (n+1). For example, adjectives, as unary functors, are modeled with 2-way tensors, or matrices. Tensor by vector multiplication formalizes function application and serves as the general composition method.

Baroni and Zamparelli (2010) propose a practical and empirically effective way to estimate matrices representing adjectival modifiers of nouns by linear regression from corpus-extracted examples of noun and adjective-noun vectors. Unlike the neural network approach of Socher et al. (2011; 2012), the Baroni and Zamparelli method does not require manually labeled data nor costly iterative estimation procedures, as it relies on automatically extracted phrase vectors and on the analytical solution of the least-squares-error problem.

The same method was later applied to matrix representations of intransitive verbs and determiners (Bernardi et al., 2013; Dinu et al., 2013), always with good empirical results.

The full range of semantic types required for natural language processing, including those of adverbs and transitive verbs, has to include, however, tensors of greater rank. The estimation method originally proposed by Baroni and Zamparelli has been extended to 3-way tensors representing transitive verbs by Grefenstette et al. (2013) with preliminary success. Grefenstette et al.'s method works in two steps. First, one estimates matrices of verb-object phrases from subject and subject-verb-object vectors; next, transitive verb tensors are estimated from verb-object matrices and object vectors.

1.2 Problems with the extension of the lexical function model to sentences

With all the advantages of lf, scaling it up to arbitrary sentences, however, leads to several issues. In particular, it is desirable for all practical purposes to limit representation size. For example, if noun meanings are encoded in vectors of 300 dimensions, adjectives become matrices of 300^2 cells, and transitive verbs are represented as tensors with 300^3=$27,000,000$ dimensions.

Estimating tensors of this size runs into data sparseness issues already for less common transitive verbs. Indeed, in order to train a transitive verb tensor (e.g., *eat*), the method of Grefenstette et al. (2013) requires a sufficient number of distinct verb object phrases with that verb (e.g., *eat cake*, *eat fruits*), each attested in combination with a certain number of subject nouns with sufficient frequency to extract sensible vectors. It is not feasible to obtain enough data points for all verbs in such a training design.

Things get even worse for other categories. Adverbs like *quickly* that modify intransitive verbs have to be represented with 300^{2^2} = $8,100,000,000$ dimensions. Modifiers of transitive verbs would have even greater representation size, which may not be possible to store and learn efficiently.

Another issue is that the same or similar items that occur in different syntactic contexts are assigned different semantic types with incomparable representations. For example, verbs like *eat* can be used in transitive or intransitive constructions (*children eat meat*/*children eat*), or in passive (*meat is eaten*). Since predicate arity is encoded in the order of the corresponding tensor, *eat* and the like have to be assigned different representations (matrix or tensor) depending on the context. Deverbal nouns like *demolition*, often used without mention of who demolished what, would have to get vector representations while the corresponding verbs (*demolish*) would become tensors, which makes immediately related verbs and nouns incomparable. Nouns in general would oscillate between vector and matrix representations depending on argument vs. predicate vs. modifier position (*an animal runs* vs. *this is an animal* vs. *animal shelter*). Prepositions are the hardest, as the syntactic positions in which they occur are most diverse (*park in the dark* vs. *play in the dark* vs. *be in the dark* vs. *a light glowing in the dark*).

In all those cases, the same word has to be mapped to tensors of different orders. Since each of these tensors must be learned from examples individually, their obvious relation is missed. Besides losing the comparability of the semantic contribution of a word across syntactic contexts, we also worsen the data sparseness issues.

The last, and related, point is that for the tensor calculus to work, one needs to model, for each word, each of the constructions in the corpus that the word is attested in. In its pure form lf does not include an emergency backoff strategy when unknown words or constructions are encountered. For example, if we only observe transitive usages of *to eat* in the training corpus, and encounter an intransitive or passive example of it in testing data,

the system would not be able to compose a sentence vector at all. This issue is unavoidable since we don't expect to find all words in all possible constructions even in the largest corpus.

2 The practical lexical function model

As follows from section 1.2, it would be desirable to have a compositional distributional model that encodes function-argument relations but avoids the troublesome high-order tensor representations of the pure lexical function model, with all the practical problems that come with them. We may still want to represent word meanings in different syntactic contexts differently, but at the same time we need to incorporate a formal connection between those representations, e.g., between the transitive and the intransitive instantiations of the verb *to eat*. Last but not least, all items need to include a common aspect of their representation (e.g., a vector) to allow comparison across categories (the case of *demolish* and *demolition*).

To this end, we propose a new model of composition that maintains the idea of function application, while avoiding the complications and rigidity of lf. We call our proposal *practical lexical function* model, or *plf*. In plf, a functional word is not represented by a single tensor of arity-dependent order, but by a vector plus an ordered set of matrices, with one matrix for each argument the function takes. After applying the matrices to the corresponding argument vectors, a single representation is obtained by summing across all resulting vectors.

2.1 Word meaning representation

In plf, all words are represented by a vector, and functional words, such as predicates and modifiers, are also assigned one or more matrices. The general form of a semantic representation for a linguistic unit is an ordered tuple of a vector and $n \in \mathbb{N}$ matrices:[1]

$$\left\langle \vec{x}, \overset{\square}{x}_1, \ldots, \overset{\square}{x}_n \right\rangle$$

The number of matrices in the representation encodes the arity of a linguistic unit, i.e., the number of other units to which it applies as a function. Each matrix corresponds to a function-argument relation, and words have as many matrices as many arguments they take: none for (most) nouns, one for adjectives and intransitive verbs, two for transitives, etc. The matrices formalize argument slot saturation, operating on an argument vector representation through matrix by vector multiplication, as described in the next section.

[1] Matrices associated with term x are symbolized $\overset{\square}{x}$.

dog	$\vec{dog}$
run	$\vec{run}, \overset{\square}{run}$
chase	$\vec{chase}, \overset{\square_s}{chase}, \overset{\square_o}{chase}$
give	$\vec{give}, \overset{\square_s}{give}, \overset{\square_o}{give}, \overset{\square_{io}}{give}$
big	$\vec{big}, \overset{\square}{big}$
very	$\vec{very}, \overset{\square_n}{very}, \overset{\square_a}{very}$
quickly	$\vec{quickly}, \overset{\square_s}{quickly}, \overset{\square_v}{quickly}$

Table 1: Examples of word representations. Subscripts encode, just for mnemonic purposes, the constituent whose vector the matrix combines with: **s**ubject, **o**bject, **i**ndirect **o**bject, **n**oun, **a**djective, **v**erb phrase.

Modifiers of n-ary functors are represented by n+1-ary structures. For instance, we treat adjectives that modify nouns (0-ary) as unary functions, encoded in a vector-matrix pair. Adverbs have different semantic types depending on their syntactic role. Sentential adverbs are unary, while adverbs that modify adjectives (*very*) or verb phrases (*quickly*) are encoded as binary functions, represented by a vector and two matrices. The form of semantic representations we are using is shown in Table 1.[2]

2.2 Semantic composition

Our system incorporates semantic composition via two composition rules, one for combining structures of different arity and the other for symmetric composition of structures with the same arity. These rules incorporate insights of two empirically successful models, lexical function and the simple additive approach, used as the default structure merging strategy.

The first rule is *function application*, illustrated in Figure 1. Table 2 illustrates simple cases of function application. For transitive verbs semantic composition applies iteratively as shown in the derivation of Figure 2. For ternary predicates such

[2] To determine the number and ordering of matrices representing the word in the current syntactic context, our plf implementation relies on the syntactic type assigned to the word in the categorial grammar parse of the sentence.

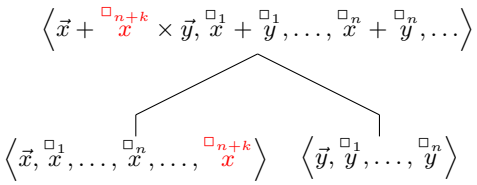

Figure 1: Function application: If two syntactic sisters have different arity, treat the higher-arity sister as the functor. Compose by multiplying the last matrix in the functor tuple by the argument vector and summing the result to the functor vector. Unsaturated matrices are carried up to the composed node, summing across sisters if needed.

dogs	$\vec{dogs}$
run	$\vec{run}, \square run$
dogs run	$\vec{run} + \square run \times \vec{dog}$
house	$\vec{house}$
big	$\vec{big}, \square big$
big house	$\vec{big} + \square big \times \vec{house}$

Table 2: Examples of function application.

as *give* in a ditransitive construction, the first step in the derivation absorbs the innermost argument by multiplying its vector by the third *give* matrix, and then composition proceeds like for transitives.

The second composition rule, *symmetric composition* applies when two syntactic sisters are of the same arity (e.g., two vectors, or two vector-matrix pairs). Symmetric composition simply sums the objects in the two tuples: vector with vector, n-th matrix with n-th matrix.

Symmetric composition is reserved for structures in which the function-argument distinction is problematic. Some candidates for such treatment are coordination and nominal compounds, although we recognize that the headless analysis is

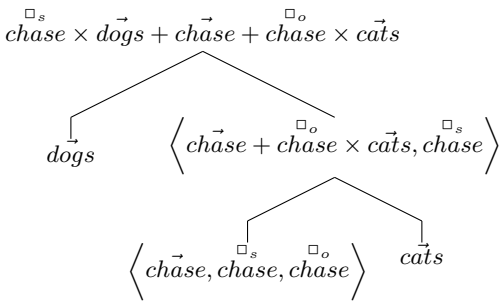

Figure 2: Applying function application twice to derive the representation of a transitive sentence.

sing: $\vec{sing}, \square sing$	dance: $\vec{dance}, \square dance$
sing and dance:	$\vec{sing} + \vec{dance}, \square sing + \square dance$
rice: $\vec{rice}$	cake: $\vec{cake}$
rice cake	$\vec{rice} + \vec{cake}$

Table 3: Examples of symmetric composition.

not the only possible one here. See two examples of Symmetric Composition application in Table 3.

Note that the *sing and dance* composition in Table 3 skips the conjunction. Our current plf implementation treats most grammatical words, including conjunctions, as "empty" elements, that do not project into semantics. This choice leads to some interesting "serendipitous" treatments of various constructions. For example, since the copula is empty, a sentence with a predicative adjective (*cars are red*) is treated in the same way as a phrase with the same adjective in attributive position (*red cars*) – although the latter, being a phrase and not a full sentence, will later be embedded as argument in a larger construction. Similarly, leaving the relative pronoun empty makes *cars that run* identical to *cars run*, although, again, the former will be embedded in a larger construction later in the derivation.

We conclude our brief exposition of plf with an alternative intuition for it: the plf model is also a more sophisticated version of the additive approach, where argument words are adapted by matrices that encode the relation to their functors before the sentence vector is derived by summing.

2.3 Satisfying the desiderata

Let us now outline how plf addresses the shortcomings of lf listed in Section 1.2. First, all issues caused by representation size disappear. An n-ary predicate is no longer encoded as an n+1-way tensor; instead we have a sequence of n matrices. The representation size grows linearly, not exponentially, for higher semantic types, allowing for simpler and more efficient parameter estimation, storage, and computation.

As a consequence of our architecture, we no longer need to perform the complicated step-by-step estimation for elements of higher arity. Indeed, one can estimate each matrix of a complex representation individually using the simple method of Baroni and Zamparelli (2010). For instance, for transitive verbs we estimate the verb-subject combination matrix from subject and verb-

boys	$\vec{boys}$
eat (intrans.)	$\vec{eat}, \overset{\square_s}{eat}$
boys eat	$\overset{\square_s}{eat} \times \vec{boys} + \vec{eat}$
meat	$\vec{meat}$
eat (trans.)	$\vec{eat}, \overset{\square_s}{eat}, \overset{\square_o}{eat}$
boys eat meat	$\overset{\square_s}{eat} \times \vec{boys} + \vec{eat} + \overset{\square_o}{eat} \times \vec{meat}$
(is) eaten (pass.)	$\vec{eat}, \overset{\square_o}{eat}$
meat is eaten	$\vec{eat} + \overset{\square_o}{eat} \times \vec{meat}$

Table 4: The verb *to eat* associated to different sets of matrices in different syntactic contexts.

subject vectors, the verb-object combination matrix from object and verb-object vectors. We expect a reasonably large corpus to feature many occurrences of a verb with a variety of subjects and a variety of objects (but not necessarily a variety of subjects with each of the objects as required by Grefenstette et al.'s training), allowing us to avoid the data sparseness issue.

The semantic representations we propose include a semantic vector for constituents of any semantic type, thus enabling semantic comparison for words of different parts of speech (the case of *demolition* vs. *demolish*).

Finally, the fact that we represent the predicate interaction with each of its arguments in a separate matrix allows for a natural and intuitive treatment of argument alternations. For instance, as shown in Table 4, one can distinguish the transitive and intransitive usages of the verb *to eat* by the presence of the object-oriented matrix of the verb while keeping the rest of the representation intact. To model passive usages, we insert the object matrix of the verb only, which will be multiplied by the syntactic subject vector, capturing the similarity between *eat meat* and *meat is eaten*.

So keeping the verb's interaction with subject and object encoded in distinct matrices not only solves the issues of representation size for arbitrary semantic types, but also provides a sensible built-in strategy for handling a word's occurrence in multiple constructions. Indeed, if we encounter a verb used intransitively which was only attested as transitive in the training corpus, we can simply omit the object matrix to obtain a type-appropriate representation. On the other hand, if the verb occurs with more arguments than usual in testing materials, we can add a default diagonal identity matrix to its representation, signaling agnosticism about how the verb relates to the unexpected argument. This flexibility makes our model suitable to compute vector representations of sentences without stumbling at unseen syntactic usages of words.

To summarize, plf is an extension of the lexical function model that inherits its strengths and overcomes its weaknesses. We still employ a linguistically-motivated notion of semantic composition as function application and use distinct kinds of representations for different semantic types. At the same time, we avoid high order tensor representations, produce semantic vectors for all syntactic constituents, and allow for an elegant and transparent correspondence between different syntactic usages of a lexeme, such as the transitive, the intransitive, and the passive usages of the verb *to eat*. Last but not least, our implementation is suitable for realistic language processing since it allows to produce vectors for sentences of arbitrary size, including those containing novel syntactic configurations.

3 Evaluation

3.1 Evaluation materials

We consider 5 different benchmarks that focus on different aspects of sentence-level semantic composition. The first data set, created by Edward Grefenstette and Mehrnoosh Sadrzadeh and introduced in Kartsaklis et al. (2013), features 200 sentence pairs that were rated for similarity by 43 annotators. In this data set, sentences have fixed adjective-noun-verb-adjective-noun (anvan) structure, and they were built in order to crucially require context-based verb disambiguation (e.g., *young woman filed long nails* is paired with both *young woman smoothed long nails* and *young woman registered long nails*). We also consider a similar data set introduced by Grefenstette (2013), comprising 200 sentence pairs rated by 50 annotators. We will call these benchmarks **anvan1** and **anvan2**, respectively. Evaluation is carried out by computing the Spearman correlation between the annotator similarity ratings for the sentence pairs and the cosines of the vectors produced by the various systems for the same sentence pairs.

The benchmark introduced by The Pham et al. (2013) at the TFDS workshop (**tfds** below) was specifically designed to test compositional methods for their sensitivity to word order and the semantic effect of determiners. The tfds benchmark contains 157 target sentences that are matched with a set of (approximate) paraphrases (8 on av-

erage), and a set of "foils" (17 on average). The foils have high lexical overlap with the targets but very different meanings, due to different determiners and/or word order. For example, the target *A man plays an acoustic guitar* is matched with paraphrases such as *A man plays guitar* and *The man plays the guitar*, and foils such as *The man plays no guitar* and *A guitar plays a man*. A good system should return higher similarities for the comparison with the paraphrases with respect to that with the foils. Performance is assessed through the t-standardized cross-target average of the difference between mean cosine with paraphrases and mean cosine with foils (Pham and colleagues, equivalently, reported non-standardized average and standard deviations).

The two remaining data sets are larger and more 'natural', as they were not constructed by linguists under controlled conditions to focus on specific phenomena. They are aimed at evaluating systems on the sort of free-form sentences one encounters in real-life applications. The **msrvid** data set from the SemEval-2012 Semantic Textual Similarity (STS) task (Agirre et al., 2012) consists of 750 sentence pairs that describe brief videos. Sentence pairs were scored for similarity by 5 subjects each. Following standard practice in paraphrase detection studies (e.g., Blacoe and Lapata (2012)), we use cosine similarity between sentence pairs as computed by one of our systems together with two shallow similarity cues: word overlap between the two sentences and difference in sentence length. We obtain a final similarity score by weighted addition of the 3 cues, with the optimal weights determined by linear regression on separate msrvid train data that were also provided by the SemEval task organizers (before combining, we checked that the collinearity between cues was low). System scores are evaluated by their Pearson correlation with the human ratings.

The final set we use is **onwn**, from the *SEM-2013 STS shared task (Agirre et al., 2013). This set contains 561 pairs of glosses (from the WordNet and OntoNotes databases), rated by 5 judges for similarity. Our main interest in this set stems from the fact that glosses are rarely well-formed full sentences (consider, e.g., *cause something to pass or lead somewhere*; *coerce by violence, fill with terror*). For this reason, they are very challenging for standard parsers. Indeed, we estimated from a sample of 40 onwn glosses that the C&C parser (see below) has only 45% accuracy on this set. Since *plf* needs syntactic information to construct sentence vectors compositionally, we test it on onwn to make sure that it is not overly sensitive to parser noise. Evaluation proceeds as with msrvid (cue weights are determined by 10-fold cross-validation).[3]

3.2 Semantic space construction and composition model implementation

Our source corpus was given by the concatenation of ukWaC (`wacky.sslmit.unibo.it`), a mid-2009 dump of the English Wikipedia (`en.wikipedia.org`) and the British National Corpus (`www.natcorp.ox.ac.uk`), for a total of about 2.8 billion words.

We collected a 30K-by-30K matrix by counting co-occurrence of the 30K most frequent content lemmas (nouns, adjectives and verbs) within a 3-word window. The raw count vectors were transformed into positive Pointwise Mutual Information scores and reduced to 300 dimensions by the Singular Value Decomposition. All vectors were normalized to length 1. This setup was picked without tuning, as we found it effective in previous, unrelated experiments.[4]

We consider four composition models. The **add** (additive) model produces the vector of a sentence by summing the vectors of all content words in it. Similarly, **mult** uses component-wise multiplication of vectors for composition. While these models are very simple, a long experimental tradition has proven their effectiveness (Landauer and Dumais, 1997; Mitchell and Lapata, 2008; Mitchell and Lapata, 2010; Blacoe and Lapata, 2012).

For the **lf** (lexical function) model, we construct functional matrix representations of adjectives, determiners and intransitive verbs. These are trained using Ridge regression with generalized cross-validation from corpus-extracted vectors of nouns,

[3] We did not evaluate on other STS benchmarks since they have characteristics, such as high density of named entities, that would require embedding our compositional models into more complex systems, obfuscating their impact on the overall performance.

[4] With the multiplicative composition model we also tried Nonnegative Matrix Factorization instead of Singular Value Decomposition, because the negative values produced by SVD are potentially problematic for mult. In addition, we repeated the evaluation for the multiplicative and additive models without any form of dimensionality reduction. The overall pattern of results did not change significantly, and thus for consistency we report all models' performance only for the SVD-reduced space.

as input, and phrases including those nouns as output (e.g., the matrix for *red* is trained from corpus-extracted $\langle noun, red\text{-}noun \rangle$ vector pairs). Transitive verb tensors are estimated using the two-step regression procedure outlined by Grefenstette et al. (2013). We did not attempt to train a lf model for the larger and more varied msrvid and onwn data sets, as this would have been extremely time consuming and impractical for all the reasons we discussed in Section 1.2 above.

Training **plf** (practical lexical function) proceeds similarly, but we also build preposition matrices (from $\langle noun, preposition\text{-}noun \rangle$ vector pairs), and for verbs we prepare separate subject and object matrices.

Since syntax guides lf and plf composition, we supplied all test sentences with categorial grammar parses. Every sentence in the anvan1 and anvan2 datasets has the form (subject) Adjective + Noun + Transitive Verb + (object) Adjective + Noun, so parsing them is trivial. All sentences in tfds have a predictable structure that allows perfect parsing with simple finite state rules. In all these cases, applying a general-purpose parser to the data would have, at best, had no impact and, at worst, introduced parsing errors. For msrvid and onwn, we used the output of the C&C parser (Clark and Curran, 2007).

3.3 Results

Table 5 summarizes the performance of our models on the chosen tasks, and compares it to the state of the art reported in previous work, as well as to various strong baselines.

The plf model performs very well on both anvan benchmarks, outperforming not only add and mult, but also the full-fledged lf model. Given that these data sets contain, systematically, transitive verbs, the major difference between plf and lf lies in their representation of the latter. Evidently, the separately-trained subject and object matrices of plf, being less affected by data sparseness than the 3-way tensors of lf, are better able to capture how verbs interact with their arguments. For anvan1, plf is just below the state of the art, which is based on disambiguating the verb vector in context (Kartsaklis and Sadrzadeh, 2013), and lf outperforms the baseline, which consists in using the verb vector only as a proxy to sentence similarity.[5] On anvan2, plf outperforms the best model reported by Grefenstette (2013) (an implementation of the lexical function ideas along the lines of Grefenstette and Sadrzadeh (2011a; 2011b)). And lf is, again, the only model, besides plf, that performs better than the baseline.

models	anvan 1	anvan 2	tfds	msr vid	onwn
add	8	22	-0.2	78	66
mult	8	-4	-2.3	77	55
lf	15	30	**5.90**	NA	NA
plf	**20**	**36**	2.7	**79**	**67**
soa	22	27	11.4	87	75
baseline	8	22	7.9	77	55

Table 5: Performance of composition models on all evaluation sets. Figures of merit follow previous art on each set and are: percentage Spearman coefficients for anvan1 and anvan2, t-standardized average difference between mean cosines with paraphrases and with foils for tfds, percentage Pearson coefficients for msrvid and onwn. State-of-the-art (soa) references: anvan1: Kartsaklis and Sadrzadeh (2013); anvan2: Grefenstette (2013); tfds: The Pham et al. (2013); msrvid: Bär et al. (2012); onwn: Han et al. (2013). Baselines: anvan1/anvan2: verb vectors only; tfds: word overlap; msrvid/onwn: word overlap + sentence length.

In the tfds task, not surprisingly the add and mult models, lacking determiner representations and being order-insensitive, fail to distinguish between true paraphrases and foils (indeed, for the mult model foils are significantly *closer* to the targets than the paraphrases, probably because the latter have lower content word overlap than the foils, that often differ in word order and determiners only). Our plf approach is able to handle determiners and word order correctly, as demonstrated by a highly significant ($p < 0.01$) difference between paraphrase and foil similarity (average difference in cosine .017, standard deviation .077). In this case, however, the traditional lf model (average difference .044, standard deviation .092) outperforms plf. Since determiners are handled identically under the two approaches, the culprit must be word order. We conjecture that the lf 3-way tensor representation of transitive verbs leads to a stronger asymmetry between sentences with in-

[5] We report state of the art from Kartsaklis and Sadrzadeh (2013) rather than Kartsaklis et al. (2013), since only the former used a source corpus that is comparable to ours.

verted arguments, and thus makes this model particularly sensitive to word order differences. Indeed, if we limit evaluation to those foils characterized by word order changes only, lf discriminates between paraphrases and foils even more clearly, whereas the plf difference, while still significant, decreases slightly.

The state-of-the-art row for tfds reports the lf implementation by The Pham et al. (2013), which outperforms ours. The main difference is that Pham and colleagues do not normalize vectors like we do. If we don't normalize, we do get larger differences for our models as well, but consistently lower performance in all other tasks. More worryingly, the simple word overlap baseline reported in the table sports a larger difference than our best model. Clearly, this baseline is exploiting the systematic determiner differences in the foils and, indeed, when it is evaluated on foils where only word order changes its performance is no longer significant.

On msrvid, the plf approach outperforms add and mult, although the difference between the three is not big. Our result stands in contrast with Blacoe and Lapata (2012), the only study we are aware of that compared a sophisticated composition model (Socher et al.'s 2011 model) to add and mult on realistic sentences, which attained the top performance with the simple models for both figures of merit they used.[6] The best 2012 STS system (Bär et al., 2012), obtained 0.87 correlation, but with many more and considerably more complex features than the ones we used here. Indeed, our simple system would have obtained a respectable 25/89 ranking in the STS 2012 msrvid task. Still, we must also stress the impressive performance of our baseline, given by the combination of the word overlap and sentence length cues. This suggests that the msrvid benchmark lacks the lexical and syntactic variety we would like to test our systems on.

Our plf model is again the best on the onwn set (albeit by a small margin over add). This is a very positive result, in the light of the fact that the parser has very low performance on the onwn glosses, thus suggesting that plf can produce sensible semantic vectors from noisy syntactic representations. Here the overlap+length baseline does not perform so well, and again the best STS 2013 system (Han et al., 2013) uses considerably richer knowledge sources and algorithms than ours. Our plf-based method would have reached a respectable 20/90 rank in the STS 2013 onwn task.

As a final remark, in all experiments the running time of plf was only slightly larger than for the simpler models, but orders of magnitude smaller than lf, confirming another practical side of our approach.

4 Conclusion

We introduced an approach to compositional distributional semantics based on a linguistically-motivated syntax-to-semantics type mapping, but simple and flexible enough that it can produce representations of English sentences of arbitrary size and structure.

We showed that our approach is competitive against the more complex lexical function model when evaluated on the simple constructions the latter can be applied to, and it outperforms the additive and multiplicative compositionality models when tested on more realistic benchmarks (where the full-fledged lexical function approach is difficult or impossible to use), even in presence of strong noise in its syntactic input. While our results are encouraging, no current benchmark combines large-scale, real-life data with the syntactic variety on which a syntax-driven approach to semantics such as ours could truly prove its worth. The recently announced SemEval 2014 Task 1[7] is filling exactly this gap, and we look forward to apply our method to this new benchmark, as soon as it becomes available.

One of the strengths of our framework is that it allows for incremental improvement focused on specific constructions. For example, one could add representations for different conjunctions (*and* vs. *or*), train matrices for verb arguments other than subject and direct object, or include new types of modifiers into the model, etc.

While there is potential for local improvements, our framework, which extends and improves on existing compositional semantic vector models, has demonstrated its ability to account for full sentences in a principled and elegant way. Our implementation of the model relies on simple and effi-

[6] We refer here to the results reported in the erratum available at `http://homepages.inf.ed.ac.uk/s1066731/pdf/emnlp2012erratum.pdf`. The add/mult advantage was even more marked in the original paper.

[7] `http://alt.qcri.org/semeval2014/task1/`

cient training, works fast, and shows good empirical results.

Acknowledgements

We thank Roberto Zamparelli and the COMPOSES team for helpful discussions. This research was supported by the ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: a pilot on semantic textual similarity. In *Proceedings of *SEM*, pages 385–393, Montreal, Canada.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Proceedings of *SEM*, pages 32–43, Atlanta, GA.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of *SEM*, pages 435–440, Montreal, Canada.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2013. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*. In press; `http://clic.cimec.unitn.it/composes/materials/frege-in-space.pdf`.

Raffaella Bernardi, Georgiana Dinu, Marco Marelli, and Marco Baroni. 2013. A relatedness benchmark to test the role of determiners in compositional distributional semantics. In *Proceedings of ACL (Short Papers)*, pages 53–57, Sofia, Bulgaria.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, pages 1394–1404, Edinburgh, UK.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of GEMS*, pages 62–66, Edinburgh, UK.

Edward Grefenstette, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of IWCS*, pages 131–142, Potsdam, Germany.

Edward Grefenstette. 2013. *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. PhD thesis, University of Oxford Essex.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of *SEM*, pages 44–52, Atlanta, GA.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of EMNLP*, pages 1590–1601, Seattle, WA.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of CoNLL*, pages 114–123, Sofia, Bulgaria.

Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809, Granada, Spain.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Nghia The Pham, Raffaella Bernardi, Yao-Zhong Zhang, and Marco Baroni. 2013. Sentence paraphrase detection: When determiners and word order make the difference. In *Proceedings of the Towards a Formal Distributional Semantics Workshop at IWCS 2013*, pages 21–29, Potsdam, Germany.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.