No Context Left Behind: Meta-Embeddings with Word Vectors Trained on Different Notions of Context

Konstantinos Christopher Tsiolis
Mila / McGill University
kc.tsiolis@mail.mcgill.ca

Abstract

Recent work has shown that combining pre-trained word embeddings from different algorithms into a single word embedding leads to improved intrinsic and extrinsic performance, even with approaches as simple as concatenating or averaging the source vectors. However, popular embedding algorithms such as word2vec, GloVe, and FastText share very similar training objectives based on cooccurrences between words in a context window of fixed size. Hence, we seek to build meta-embeddings that incorporate embeddings trained with different notions of context (such as differing context window sizes and dependency-based embeddings). We combine up to 9 embeddings from the same algorithm, Hilbert-MLE, with varying notions of context and observe superior performance to each of the individual source embeddings on 5 downstream tasks with the new meta-embeddings. We experiment with both summation and concatenation, while proposing a more sophisticated method that can be evaluated in future work.

1 Introduction

There now exist countless pre-trained word embedding algorithms, such as word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017). They have been successfully and widely used as inputs for a variety of downstream tasks (Goldberg, 2016). Recently, there has been interest in how to combine embeddings obtained from each of these algorithms to make *meta-embeddings*. This allows for the aggregation of information from multiple source embeddings, as well as to increase vocabulary coverage. Yin and Schütze (2016) find that simply concatenating source embeddings (including word2vec and GloVe) leads to improved performance on similarity, analogy, and POS tagging tasks. Coates and Bollegala (2018) show that averaging word2vec, GloVe, and hierarchical log-bilinear (Turian et al., 2010) embeddings leads to similarly promising results on similarity and analogy tasks. Kiela et al. (2018) learn context-dependent attention weights for each (word, embedding algorithm) in order to conduct a weighted sum of GloVe and FastText embeddings, and they observe small improvements in sentiment classification and natural language inference.

One issue with existing approaches for constructing meta-embeddings is that many embedding algorithms essentially have the same objective. As was shown by Newell et al. (2019), word2vec, GloVe, and FastText all belong to the class of "low rank embedders", which is to say that they bilinearly parametrize a variant of PMI between words (or characters) in a corpus. Here, we understand PMI to mean the probability of observing two words together as compared to the probability that the two words are independent. Furthermore, this paradigm alone is sufficient to construct a competitive embedding algorithm, which the authors call Hilbert-MLE. This suggests that the successes that have been observed with meta-embeddings have less to do with the diversity of algorithms and more to do with the diversity of corpora that each of the source embeddings were trained on. After all, the PMI between two words is dependent on the data distribution that they are being drawn from.

The definition of word cooccurrence is a second factor that influences PMI between two words. It specifies what we mean by "observing two words together". While manipulating this definition is not new, this had not been explored in the context of making word meta-embeddings prior to this work. In particular, we consider nine different notions of context, and use embeddings from each of them. We construct meta-embeddings that

achieve improved performance on four out of five downstream tasks when compared to the individual source embeddings.

We also investigate different methods of combining source embeddings. Specifically, we provide a comparison between the methods of summation and concatenation. We find that dimensionality reduction techniques aid in making concatenation more effective than summation, while also keeping embedding size reasonable.

2 Background and Related Work

In this section, we focus on addressing the prior work that directly motivates this work, and we provide a brief survey of the most notable word meta-embedding methods.

2.1 Word Embedding Algorithms

We focus our discussion specifically on the topic of cooccurrence-based word embedding algorithms, which is the class of algorithms dealt with in the Hilbert-MLE paper (Newell et al., 2019). These algorithms train two vector representations per word i: a *covector* (context vector) $\langle i|$ and a *vector* (word vector) $|i\rangle$. The objective of training is to have the covector-vector inner product $\langle i|j\rangle$ reflect a function of the frequency at which words i appears in the context of word j in a training corpus. This is done by iterating over each word in the training corpus and keeping counts of which words appear around it. In practice, a context window (typically of size 5 on either side of the word) determines which words are context words for a given word. Cooccurrence counts are weighted proportionally to the distance of the context word from the target word (this is called *dynamic window weighting*).

Newell et al. (2019) study the loss functions of multiple cooccurrence-based algorithms (including word2vec, GloVe, and FastText), and find that they all lead to an optimum where $\langle i|j\rangle$ approximates a function of $PMI(i, j)$. For example, in the case of word2vec, they find that $\langle i|j\rangle \approx PMI(i, j) - \log k$. Here, PMI (pointwise mutual information) is calculated as the probability of observing word i in the context of word j in a corpus (in the way we defined previously) divided by the product of the unigram probabilities of words i and j. Hence, we have a sense of how often two words appear together compared to how often we would expect to see each of them individually. Then, following the distributional hypothesis of (Harris, 1954), which states that similar words occur in similar contexts, the cosine distance between word vectors $|i\rangle \cdot |j\rangle$ (assuming vectors have unit norm) can be used as a measure of how "similar" two words are, at least in the opinion of a particular embedding algorithm trained on a particular corpus with a particular notion of context.

The authors then go on to construct their own embedding algorithm, Hilbert-MLE, relying solely on the principle of parametrization of PMI with covectors and vectors (as well as a gradient tempering measure). They show that this is sufficient to train embeddings which are competitive with word2vec and GloVe on similarity, analogy, and five downstream evaluations. Outside of the gradient tempering hyperparameter, it does not incorporate any of the hyperparameters present in other embedders (such as the number of negative samples in word2vec). Hence, for the purposes of our work, Hilbert-MLE is our embedder of choice.

2.2 Different Notions of Context

Multiple notions of context have been used to train word embeddings outside of the standard symmetric window of size 5 around a target word. Levy and Goldberg (2014) introduce *dependency-based* word embeddings, where the context of each target word is defined as its head and its modifiers in the dependency parse tree of a sentence, as well as the labels of the dependency relations between words. Through their qualitative analysis, they show that using a dependency-based notion of context leads to word vectors that exhibit *similarity* (rather than *relatedness*, which tends to be exhibited by large symmetric context windows). That is, the vector for a given word tends to have very high dot products with words that can directly replace it in a sentence, rather than words that deal with the same topic. For instance, the dependency-based vector for "hogwarts" is closest to vectors for "sunnydale" and "collinwood", which are names of other famous schools. In the case of word2vec with window size 5, "hogwarts" is closest to "dumbledore" and "hallows". This work reflects how simply changing the notion of context drastically alters the resulting word embeddings.

Inspired by this, our previous work sought to train Hilbert-MLE embeddings with *asymmetric* notions of context. We first attempted using *left-sided* and *right-sided* context windows, where we

only place the context window on one side of the target word. This produced markedly different interactions between covectors and vectors, but changes in vector-vector (cosine similarity) interactions were not as pronounced. We then trained a variant of depenency-based embeddings, where the only context for a target word is its head in the dependency parse tree of the sentence it appears in. We made similar observations to those of Levy and Goldberg (2014). We elaborate further on the performance of asymmetric embeddings (both individually and as source embeddings to the meta-embeddings) in Sections 5 and 6. We also extend our previous work by experimenting with additional notions of context, such as varying the context window size and incorporating subword contexts with the help of FastText (Bojanowski et al., 2017).

2.3 Meta-embeddings

Yin and Schütze (2016) present meta-embeddings as a computationally inexpensive way of obtaining high-quality word embeddings when compared to emerging deep learning methods. It allows for the pooling together of existing pre-trained word embeddings trained on a wide variety of corpora with a wide variety of algorithms. Training time for meta-embeddings is minimal, especially if embeddings are simply concatenated. The authors experiment with concatenation, singular value decomposition (to reduce the dimensionality of the concatenation operation), and a neural method called *1toN*. The latter consists of training randomly initialized meta-embeddings by learning maps that project them down to each of the source embeddings. All methods outperform the individual source embeddings on similarity, analogy, and POS tagging tasks. Concatenation performs quite well, but suffers from high dimensionality and is hurt by dimensionality reduction.

Coates and Bollegala (2018) provide a mathematical treatment of the concatenation and the averaging of word embeddings, showing why each can be effective in producing meta-embeddings. Both techniques are capable of preserving the notion of distance between words (as specified by each of the source embeddings). This is the case for concatenation because it inherently guarantees orthogonality between source embeddings. Averaging does not have this guarantee, but instead relies on the phenomenon that vectors originating from different source embedding sets are likely to be orthogonal to each other, which the authors observe empiricially. The authors show that averaging is competitive with concatenation on similarity and relatedness tasks, while benefitting from small dimensionality.

Kiela et al. (2018) propose *dynamic meta-embeddings* (DME), which are the result of a weighted linear combination of source embeddings. Attention weights are learned for this linear combination, thus allowing for more attention to be placed on a particular source embedding depending on the task that the meta-embeddings are being used for. They further enhance the attention mechanism by proposing a contextualized version, where attention weights are learned via a BiLSTM that takes the input sequence as context. The authors evaluate DMEs on the downstream tasks of sentiment classification and natural language inference, but only observe marginal improvements over the concatenation baseline. However, it should be noted that FastText and GloVe were the only source embeddings that were used.

Though all of the above methods achieve small gains in their respective evaluations, they essentially combine embeddings that have the same objective. Furthermore, the best way to combine embeddings in order to form meta-embeddings is still an open question. In this work, we shed light on these issues by constructing meta-embeddings with source embeddings trained on different notions of context, and by our evaluation and discussion of methods to form meta-embeddings.

3 Method

We first introduce some notation that will be useful for the following subsections. Suppose that we have n distinct collections of source embeddings. We call these collections *facets*. Each facet k has a vocabulary $\mathcal{V}_k$, and we take our meta-embedding vocabulary to be $\mathcal{V} = \bigcup_{k=1}^{n} \mathcal{V}_k$. Let d_k denote the dimension of the vectors in facet k, $k \in \{1, 2, ..., n\}$, and d denote the target dimension for the meta-embeddings. For each word i in $\mathcal{V}$, we seek to construct a meta-embedding $|i\rangle = f(|i\rangle_1, |i\rangle_2, ..., |i\rangle_n)$, where $|i\rangle_k$, $k \in \{1, 2, ..., n\}$, is the embedding for word i under facet k, and $f : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \times ... \times \mathbb{R}^{d_n} \rightarrow \mathbb{R}^d$ is a function from the source embeddings to the meta-embedding. f is the mechanism by which we build meta-embeddings from source embeddings.

In this work, we consider the cases where f is addition and where f is concatenation.

Naturally, it is possible that $i \notin \bigcap_{k=1}^{n} \mathcal{V}_k$. In this case, we simply set $|i\rangle_k = 0$ for all k such that $i \notin \mathcal{V}_k$. This seems the most natural choice as we will be summing and concatenating source embeddings, and we leave a more detailed approach for handling OOV items to future work.

3.1 Summation

Assume without loss of generality that $d_1 = d_2 = ... = d_n = d$. (If not, then in the case $d_k < d$, we simply zero-pad the vectors in facet k, and in the case $d_k > d$, we project the vectors in facet k down to dimension d via principal component analysis). Then, for a word i, we have that:

$$|i\rangle = |i\rangle_1 + |i\rangle_2 + ... + |i\rangle_n \tag{1}$$

As is noted by Coates and Bollegala (2018), there is no correspondence between the dimensions of each of the source embeddings. Hence, if we simply consider a meta-embedding for a single word, it is difficult to interpret. In fact, the dimensions of the individual source embeddings are already difficult to interpret (Faruqui et al., 2015).

As is often the case with word embeddings, we instead consider the interactions between word vectors to illustrate how summation can be effective, in very similar fashion to (Coates and Bollegala, 2018). For two words $i, j \in \mathcal{V}$, we consider the cosine similarity between their meta-embeddings (once again assuming meta-embeddings have unit norm):

$$\begin{aligned} |i\rangle \cdot |j\rangle &= (|i\rangle_1 + |i\rangle_2 + ... + |i\rangle_n) \\ &\quad \cdot (|j\rangle_1 + |j\rangle_2 + ... + |j\rangle_n) \\ &= \sum_{k=1}^{n} \sum_{l=1}^{n} |i\rangle_k \cdot |j\rangle_l \end{aligned} \tag{2}$$

We can see from the above that the dot product $|i\rangle \cdot |j\rangle$ is polluted by dot products between embeddings from different facets ($|i\rangle_k \cdot |j\rangle_l$, when $k \neq l$). The latter products do not carry any meaning. Fortunately, these products are expected to be close to zero, as was discussed in Coates and Bollegala (2018). However, similarly to the famous birthday paradox of probability, this assumption will break down as the number of facets increases (Kenyon-Dean, 2019).

3.2 Concatenation

Concatenation resolves the orthogonality issue mentioned above, since source embeddings are orthogonal in the meta-embeddings space by construction. In the case of concatenation, input vectors do not have to be transformed to the same dimension. Here, for a word i, we simply have that:

$$|i\rangle = [|i\rangle_1, |i\rangle_2, ..., |i\rangle_n] \tag{3}$$

Now, if we consider the cosine similarity between meta-embeddings for words $i, j \in \mathcal{V}$:

$$\begin{aligned} |i\rangle \cdot |j\rangle &= [|i\rangle_1, |i\rangle_2, ..., |i\rangle_n] \\ &\quad \cdot [|j\rangle_1, |j\rangle_2, ..., |j\rangle_n] \\ &= \sum_{k=1}^{n} |i\rangle_k \cdot |j\rangle_k \end{aligned} \tag{4}$$

Here, we observe that the meaningless dot products that were present in the case of summation are not present here. Rather, the cosine similarity between meta-embeddings is essentially an aggregation of the "opinions" of each of the source embeddings on how close together the two words should be. In this case, each "opinion" is weighted equally, but we can also consider a case where each facet has an associated attention weight. We intend on exploring this in future work.

Naturally, the meta-embedding dimension $d = d_1 + d_2 + ... + d_n$ increases as the number of facets n increases, quickly making concatenation impractical. Similarly to (Yin and Schütze, 2016), we use a dimensionality reduction technique to counter this issue. Rather than singular value decomposition, we use principal component analysis (PCA) on the meta-embeddings that result from concatenation.

4 Experiments

We design experiments with two goals in mind. First, we investigate the performance of meta-embeddings constructed from embeddings trained of different notions of context. Second, we investigate two methods of constructing meta-embeddings (addition and concatenation).

4.1 Contexts

We consider the following notions of context, all of which are used to train Hilbert-MLE embeddings:

- **Win1:** Context window of size 1 (on either side of the target word).

- **Win2:** Context window of size 2 (on either side of the target word).
- **Win5:** Context window of size 5 (on either side of the target word).
- **Win10:** Context window of size 10 (on either side of the target word).
- **Win20:** Context window of size 20 (on either side of the target word).
- **Far:** Context window that includes all words at a distance between 20 and 30 of the target word (on either side of the target word).
- **Deps:** Dependency-based embeddings (where the only context word of a given target word is its head).
- **Left:** Context window of size 5 (to the left of the target word).
- **Right:** Context window of size 5 (to the right of the target word).

As in Levy and Goldberg (2014), we expect that narrower windows will prioritize similarity between words, while broader windows will prioritize relatedness.

Hilbert-MLE embeddings are trained on the same corpus as in Newell et al. (2019): a combination of English Gigaword 3 (Graff et al., 2007) and a lowercased Wikipedia 2018 dump consisting of 5.4 billion tokens. The only exception to this is *Deps*, which uses the dependency parse obtained by running Stanford CoreNLP (Manning et al., 2014) on the English Gigaword 3 corpus. In both cases, we take the top 50,000 words as the vocabulary (we intend on experimenting with a vocabulary size of 500,000 in the future). We use the sample-based implementation of Hilbert-MLE [1] and set the embedding dimension to 300.

We then consider 4 different combinations of Hilbert-MLE embeddings. For each combination, we take the union of the vocabularies for each of the embeddings involved (the vocabularies are the same for all embeddings except Deps) and then take the sum of the embeddings. The combinations are:

- **Meta 1:** Win1 + Win2 + Win5 + Win10
- **Meta 2:** Win1 + Win2 + Win5 + Win10 + Deps
- **Meta 3:** Win1 + Win2 + Win5 + Win10 + Deps + Left + Right
- **Meta 4:** Win1 + Win2 + Win5 + Win10 + Win20 + Far + Deps + Left + Right

Then, we also experiment with 300-dimensional FastText embeddings (Bojanowski et al., 2017) obtained online [2], with the aim of adding subword contexts to our meta-embeddings. Though FastText is a PMI-based embedding algorithm like Hilbert-MLE, it deals with character n-grams rather than words. We restrict the FastText embeddings to the top 50,000 vocabulary words for insertion into the best-performing meta-embeddings.

4.2 Addition versus Concatenation

Given the experiments detailed in the above subsection, we take the best-performing meta-embeddings on the downstream experiments. We then compare the performance of summation (**Sum**) to the performance of concatenation. We consider the cases of concatenation without dimensionality reduction (**Conc**), concatenation followed by PCA (**ConcProj**), and projection of the source embeddings followed by concatenation (**ProjConc**). With this, not only can we compare summation against concatenation, but we can also study the effect of dimensionality reduction on performance, as well as whether or not dimensionality reduction should be conducted before or after concatenation.

Our implementation of both addition and concatenation is available online [3].

4.3 Evaluations

we conduct both intrinsic and extrinsic evaluations. We use the same similarity/relatedness tasks that were used by Newell et al. (2019) as well the same five downstream evaluations. The similarity/relatedness tasks are: Baker Verbs 143 (B143) (Baker et al., 2014), the MEN development set (MENd) and test set (MENt) (Bruni et al., 2012), Radinsky Mechanical Turk (RMT) (Radinsky et al., 2011), Rare Words (RARE) (Luong et al., 2013), SemEval 2017 Task 2 (SE17) (Camacho-Collados et al., 2017), Simlex999 (Hill

[1] https://github.com/enewe101/hilbert

[2] https://fasttext.cc/docs/en/english-vectors.html

[3] https://github.com/kylie-box/word_prism

et al., 2015), Wordsim353 (Finkelstein et al., 2001), divided into similarity (WS-S) and relatedness (WS-R) (Agirre et al., 2009), and Yang and Powers Verbs 130 (Yang and Powers, 2006). For all tasks, we report the Spearman's ρ between human rankings of similarity (or relatedness) between words and the rankings given by cosine similarity between word embeddings. This calculation only takes into account words that are covered by the embedding vocabulary.

Downstream tasks are split into two categories: text classification and sequence labelling. In the area of text classification, we have the IMDB movie reviews dataset for sentiment analysis (IMDB) (Maas et al., 2011) and the AG-News news classification dataset (AG), preprocessed by Kenyon-Dean et al. (2018). The downstream model is the same BiLSTM-max sequence encoder (Conneau et al., 2017) used by Newell et al. (2019). For sequence labelling, we consider the task of supersense tagging (SST) (Ciaramita and Altun, 2006) on the Semcor 3.0 corpus (Miller et al., 1993), as well as part-of-speech tagging on the Wall Street Journal corpus (WSJ) and the Brown corpus (Brown). For these tasks, the BiLSTM of Huang et al. (2015) is used, with the same configuration as in the Hilbert-MLE paper. We report accuracy for all tasks, except for supersense tagging, where we report micro F1 score. For all downstream tasks, we include a random baseline which consists of randomly initialized embeddings (each dimension is sampled from the uniform distribution between -0.2 and 0.2) for the top 50,000 words in Gigaword + Wikipedia.

5 Results

We first observe in Table 1 that the meta-embeddings we constructed via summation of Hilbert-MLE facets do not lead to improved performance in similarity/relatedness tasks. In fact, all four meta-embeddings have a worse average performance than the majority of the facets. A window size of 10 has the best average performance, and is especially strong in tasks emphasizing relatedness, such as WS-R and RMT. However, it should be noted that it also achieves the best performance in WS-S, which is meant to emphasize similarity.

Unlike in the intrinsic tests, the meta-embeddings outperform the individual facets on the downstream evaluations (see Table 2). This is especially pronounced in the case of supersense tagging, where the random baseline lags far behind all of the embeddings, suggesting that the choice of embedding has a much larger impact on that task. On this task, Meta 3 outperforms the best facet (Left) by 2.49 F1. We also observe small gains made by the meta-embeddings on three out of the other four tasks. Only in the AGNews classification task does a facet (Win10) outperform the meta-embeddings.

Given its performance on supersense tagging in particular and its strong performance in the other sequence labelling tasks, we consider Meta 3 as our baseline for the other experiments. We rename it **Sum**.

In Table 3, we observe that FastText far outperforms all other embeddings in the similarity/relatedness tasks. It achieves an average score of 0.646, when compared to 0.579 for Win10, the previous best-performing facet. Furthermore, adding it to Sum worsens average performance. However, FastText performs worse than Win10 on all 5 downstream tasks. Adding FastText to Sum increases performance in sentiment classification, but decreases supersense tagging performance.

Finally, Table 5 shows that concatenation is far more effective in the similarity/relatedness tasks than summation is. Average Spearman's ρ for concatenation is 0.592, when compared to 0.527 for summation. However, since 7 embeddings are being concatenated in Conc, the resulting meta-embedding has a dimension of 2100. For a fairer comparison, we first project the 7 source embeddings down to dimension 43 with PCA and then concatenate them (obtaining a final dimension of 301) (ProjConc). We also attempt projecting Conc down to dimension 300 with PCA (ConcProj). The latter approach for dimensionality reduction obtains superior average performance in similarity tasks (0.584 for ConcProj, compared to 0.565 for ProjConc).

In downstream experiments (Table 6), the concatenation approach leads to an F1 of 70.04, which is an improvement of 1.78 over summation and 4.27 over the best-performing facet (Left). In fact, concatenation has the best performance on all three sequence labelling tasks, but it is worse than summation on the two classification tasks. In addition, ConcProj outperforms ProjConc on 3 out of 5 downstream tasks and is competitive with Conc, in spite of the much lesser meta-embedding dimen-

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
Win1	0.502	0.398	0.638	0.409	**0.651**	0.545	0.390	0.692	0.398	0.642	0.525
Win2	0.514	**0.410**	0.653	0.498	0.646	0.593	0.389	0.718	**0.473**	0.660	0.555
Win5	0.538	0.400	0.725	0.529	0.648	0.651	0.370	0.695	0.441	0.701	0.570
Win10	0.525	0.377	0.723	**0.594**	0.630	**0.717**	0.329	**0.738**	0.437	0.717	**0.579**
Win20	0.481	0.317	**0.726**	0.569	0.611	0.685	0.291	0.628	0.435	**0.724**	0.547
Far	0.378	0.212	0.678	0.529	0.548	0.670	0.268	0.574	0.251	0.658	0.476
Deps	0.493	0.402	0.689	0.520	0.617	0.623	0.302	0.694	0.348	0.654	0.534
Left	**0.542**	0.401	0.697	0.499	0.654	0.628	0.303	0.697	0.327	0.679	0.542
Right	0.514	0.391	0.687	0.569	0.623	0.623	0.284	0.697	0.468	0.667	0.530
Meta 1	0.508	0.390	0.686	0.505	0.641	0.643	0.384	0.689	0.333	0.671	0.545
Meta 2	0.421	0.393	0.686	0.493	0.647	0.642	0.361	0.673	0.312	0.667	0.529
Meta 3	0.414	0.392	0.675	0.476	0.633	0.619	**0.410**	0.659	0.336	0.662	0.527
Meta 4	0.409	0.369	0.696	0.519	0.625	0.642	0.379	0.666	0.325	0.674	0.530

Table 1: Results on the similarity/relatedness tests used for Hilbert evaluation, measured with Spearman's ρ. Best is in bold.

	IMDB	News	SST	WSJ	Brown
Random	89.52	77.73	48.25	94.68	95.94
Win1	89.84	81.04	65.76	96.43	97.63
Win2	90.08	80.98	65.61	96.54	97.58
Win5	90.52	80.89	65.75	96.48	97.62
Win10	90.28	**82.11**	65.22	96.38	97.53
Win20	89.48	81.81	65.00	96.31	97.52
Far	88.96	81.80	60.92	95.90	97.00
Deps	89.16	80.86	63.75	96.47	97.28
Left	90.04	81.13	65.77	96.41	97.52
Right	90.00	81.56	65.02	96.44	97.57
Meta 1	90.12	81.39	67.58	96.56	97.73
Meta 2	**90.56**	81.84	68.06	96.66	97.72
Meta 3	89.92	81.74	**68.26**	**96.71**	97.75
Meta 4	90.04	81.53	67.85	**96.71**	**97.77**

Table 2: Downstream task validation results for facets and meta-embeddings. Best is in bold. Best facet result is underlined. Micro F1 is reported for SST, while accuracy is reported for the other tasks.

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
Win10	0.525	0.377	0.723	0.594	0.630	0.717	0.329	0.738	0.437	0.717	0.579
FastText	0.594	0.451	0.793	0.685	0.675	0.712	0.463	0.815	0.476	0.795	**0.646**
Sum	0.414	0.392	0.675	0.476	0.633	0.619	0.410	0.659	0.336	0.662	0.527
SumFast	0.370	0.397	0.680	0.474	0.604	0.621	0.405	0.651	0.330	0.663	0.519

Table 3: Similarity/relatedness results for FastText and SumFast compared to baselines, measured with Spearman's ρ. Best average is in bold. Here, SumFast is the sum of Meta 3 and FastText.

sion.

6 Discussion

The results in Table 2 suggest that combining embeddings trained on different notions of context is beneficial for downstream task performance. This is most clearly illustrated by performance on the supersense tagging task, where meta-embeddings far outperform any individual facet. Another reason that this task is the most notable is that this is the task where the word embeddings make the most significant impact. The micro F1 score for

	IMDB	News	SST	WSJ	Brown
Win10	90.28	**82.11**	65.22	96.38	97.53
FastText	90.16	76.81	63.78	95.53	97.04
Sum	89.92	81.74	**68.26**	**96.71**	97.75
SumFast	**90.44**	81.63	67.89	96.66	**97.81**

Table 4: Downstream task validation results for FastText and SumFast. Best is in bold. Micro F1 is reported for SST, while accuracy is reported for the other tasks.

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
Win10	0.525	0.377	0.723	0.594	0.630	0.717	0.329	0.738	0.437	0.717	0.579
Sum	0.414	0.392	0.675	0.476	0.633	0.619	0.410	0.659	0.336	0.662	0.527
Conc	0.473	0.405	0.747	0.569	0.698	0.689	0.414	0.731	0.464	0.732	**0.592**
ProjConc	0.434	0.339	0.730	0.559	0.659	0.684	0.401	0.711	0.422	0.712	0.565
ConcProj	0.454	0.381	0.748	0.564	0.681	0.691	0.407	0.734	0.442	0.735	<u>0.584</u>

Table 5: Similarity/relatedness results for variants of concatenation, measured with Spearman's ρ. Best average is in bold. Second-best is underlined.

	IMDB	News	SST	WSJ	Brown
Win10	<u>90.28</u>	**82.11**	65.22	96.38	97.53
Sum	89.92	<u>81.74</u>	68.26	96.71	97.75
Conc	89.72	81.00	**70.04**	**96.77**	**97.90**
ProjConc	89.72	81.68	66.86	<u>96.72</u>	97.79
ConcProj	**90.52**	81.35	<u>68.47</u>	96.69	<u>97.81</u>

Table 6: Downstream task validation results for variants of concatenation. Best is in bold. Second-best is underlined. Micro F1 is reported for SST, while accuracy is reported for the other tasks.

randomly initialized embeddings is 17.52 below that of Left. We note that combining the window sizes between 1 and 10 already leads to increased performance on SST (67.58 F1 compared to 65.77 F1 for Left). Adding syntactic context with Deps and asymmetric contexts with Left and Right provide additional gains. However, Win20 and Far prove to be detrimental to SST performance. This is likely due to the tendency of these windows to capture broad topical information, which is not as useful for a sequence labelling task where every word must be tagged. This tendency is illustrated by the performance of the broad context windows on tasks that emphasize relatedness, such as WS-R, and RMT (where they perform better than all other facets except Win10). By contrast, these windows perform poorly on tasks emphasizing similarity, as is the case for S999 and Y130 (where they perform worse than all other facets). The main usefulness of the broad windows comes in the classification tasks, where their performance is solid, but not superior to a window size of 10.

We also provide a qualitative analysis, similar to the ones in Newell et al. (2019) and Levy and Goldberg (2014). We investigate the behaviour of both the covectors and the vectors of a given word ("president") under different notions of context. Specifically, we fix the vector for "president" and display the covectors which have the highest dot product with it. This gives an indication of which contexts are most associated with the word president and how that changes depending on the definition of context. We also fix the covector for "president" and examine which vectors have the highest dot product with it. Finally, we consider which vectors are closest to the vector for president in the embedding space (as measured by cosine similarity).

In Table 7, we observe that changing the window size does not change the covectors that the word vector for "president" is most selective for. However, if we consider the vectors that it has the highest inner product with, there is a noticeable change as window size gets larger. While Win1 puts emphasis on words that are similar to "president" (like "leader", "mayor", and "chairman"),

Win20 puts emphasis on related words (like "presidential", "meeting", and "administration"). With dependency-based embeddings, we see words like "governor", "candidate", and "lawyer" that tend to have the same POS tag and function as "president" in a sentence. This clearly illustrates how changing the notion of context changes where vectors are located in the embedding space relative to each other. Some configurations may be more preferable than others depending on the task. For instance, as was discussed previously, we should expect larger context windows to be most useful in document classification tasks. By contrast, we should expect dependency-based embeddings to be more effective in sequence labelling tasks, especially syntactic tasks like part-of-speech tagging. This is reflected well by the results in Table 2. Meta-embeddings have all of these "opinions" of which words should be closest together available to them all at once, thus possibly explaining the success of the meta embeddings seen in Table 2.

Interestingly, adding subword contexts via FastText does not consistently improve downstream task performance. In fact, it only does so in 2 out of the 5 downstream evaluations. Furthermore, FastText embeddings lag behind Win10 embeddings on all five downstream tasks (see Table 4). This in spite of the fact that FastText is the best-performing facet by a wide margin on the similarity/relatedness tasks. This further reinforces observations that intrinsic task performance of word embeddings often does not correlate to downstream task performance (Chiu et al., 2016).

A likely cause of the poor performance of FastText embeddings is the small vocabulary size, as the FastText embeddings were trained with a much larger vocabulary size of 1 million. To test this hypothesis, we ran downstream evaluations with the full collection of 1 million FastText embeddings (**FT-Full**). The results (in Table 8) show the substantial improvement given by using the full vocabulary (over both FT-50k and Win10). As future work, we resolve to train Hilbert-MLE embeddings with a vocabulary size of 500,000 and to use the full collection of FastText embeddings to construct meta-embeddings.

On the issue of summation and concatenation, the latter emerges as the clear winner. Once again, the supersense tagging task provides the best evidence for this (Table 6). However, we also note gains in the similarity/relatedness tasks. While summation actually leads to a decrease in performance compared to the individual facets, concatenation outperforms Win10, the best of the facets. Even after projection (ConcProj), the concatenation method retains good performance on both intrinsic and extrinsic evaluations.

One of the reasons for the success of concatenation might be the orthogonality that it explicitly enforces between the facet embeddings that comprise the meta-embeddings, as discussed in Section 3. Certainly this is a factor in the similarity/relatedness tasks, which directly rely on the vector-vector dot product. Summation results in this product being polluted by meaningless dot products between embeddings from different facets, which is not the case with concatenation. This problem is exacerbated in our case since we sum embeddings from 4 to 9 facets. In previous work on averaging meta-embeddings, the number of facets was limited to 3 (Coates and Bollegala, 2018).

What is more questionable is whether or not orthogonality is also a factor affecting downstream task performance. The downstream recurrent neural networks do not explicitly take dot products between word vectors. Rather, at a given time step, they take a single vector as input and feed it into the LSTM cell. However, since we are concatenating, this vector can be viewed as n vectors, where n is the number of facets. By concatenating, we are increasing the number of features that the LSTM has access to. If we also project the concatenated embeddings, then a determination is made as to which features are the most important by selecting those which have the highest variance. This can also explain why first concatenating and then projecting is preferable, rather than first projecting and then concatenating.

Summation is more difficult to interpret in the downstream setting than concatenation. Since there is no correspondence between the dimensions of two distinct embedding facets, it is not immediately clear why directly summing facets provides downstream performance gains over the individual facets.

7 Future Work

7.1 Additional Contexts

The positive result of meta-embeddings from facets with different contexts encourages future

Context	$\text{argmax}_i \langle i \vert president \rangle$	$\text{argmax}_i \langle president \vert i \rangle$	$\text{argmax}_i \vert i \rangle \cdot \vert president \rangle$
Win1	vice, george, saddam, clinton, bush,	vice, clinton, george, bush, russian	leader, mayor, chairman, governor, vice
Win2	vice, w., george, clinton, bill	vice, w., george, bush, saddam	vice, leader, chairman, premier, prime
Win5	vice, w., clinton, george, al	vice, w., clinton, george, bush	vice, presidential, administration, chairman, leadership
Win10	vice, w., president, bush, clinton	vice, w., president, clinton, george	vice, presidential, administration, chairman, leadership
Win20	vice, w., president, clinton, bush	vice, president, w., bush, clinton	presidential, vice, leaders, meeting, administration
Far	presidential, president, clinton, vice, secretary	presidential, president, vice, clinton, senate	presidential, political, vice, secretary, leaders
Left	vice, elected, former, senior, iraqi	w., clinton, bush, george, bill	vice, prime, government, chairman, minister
Right	w., clinton, bush, george, saddam	vice, elected, former, iraqi, appointed	vice, leaders, leader, candidate, presidential
Deps	clinton, milosevic, bush, yeltsin, hussein	vice, association, executive, marketing, senior	vice, governor, candidate, lawyer, chairman

Table 7: An illustration of how the covector and vector for the word "president" interact with covectors and vectors for other words depending on the chosen notion of context.

	IMDB	News	SST	WSJ	Brown
Win10	90.28	**82.11**	65.22	96.38	97.53
FT-50k	90.16	76.81	63.78	95.53	97.04
FT-Full	**90.92**	81.90	**66.59**	**96.57**	**98.05**

Table 8: Downstream task results for the 1 million FastText embeddings (compared to restricting the vocabulary to 50,000 words, as is the case for the Hilbert-MLE embeddings).

work on the addition of more contexts into meta-embeddings. As is standard for pre-trained word embeddings, we have focused on distributional contexts in this work. However, we can also consider knowledge graph contexts from ConceptNet (Speer et al., 2017), as was suggested by (Kenyon-Dean, 2019). In ConceptNet, words are nodes in a knowledge graph and are connected to each other via labelled edges. For instance, the phrase "a dog has a tail" can be represented in ConceptNet by the nodes "dog" and "tail" being connected by an edge labelled with "HasA".

Speer et al. (2017) constructed word embeddings from ConceptNet by viewing a word's context as its neighbours in the knowledge graph, rather than the words that appear near it in a corpus. These embeddings are then combined with SGNS (word2vec) embeddings via a process called "retrofitting" (Faruqui et al., 2014). The objective is to obtain a new vector for each word that is as close as possible to its original SGNS vector, while also being close to the vectors of its neighbours in the knowledge graph.

We run preliminary tests on ConceptNet Numberbatch embeddings for 516,782 English words [4]. We find that they have outstanding performance on similarity/relatedness tasks, far outperforming FastText (with its full vocabulary of 1 million) (Table 9. However, it lags behind FastText on 4 of the 5 downstream tasks, even performing worse than the random baseline by over 1% on Wall Street Journal POS tagging (Table 10).

It remains to be see if adding ConceptNet Numberbatch embeddings as a facet to our meta-

[4] https://github.com/commonsense/conceptnet-numberbatch

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
FT-Full	0.523	0.450	0.789	0.650	0.632	0.705	0.463	0.810	0.507	0.791	0.632
Concept	0.637	0.627	0.871	0.765	0.722	0.719	0.494	0.844	0.755	0.863	0.730

Table 9: Similarity/relatedness results for ConceptNet Numberbatch embeddings.

	IMDB	News	SST	WSJ	Brown
FT-Full	90.92	81.90	66.59	96.57	98.05
Concept	91.24	81.89	65.50	93.37	96.92

Table 10: Downstream task results for the ConceptNet Numberbatch embeddings.

embeddings will improve performance. We first intend on training and tuning Hilbert-MLE embeddings at the larger vocabulary size of 500,000, close to that of the ConceptNet embeddings. From there, we can construct new meta-embeddings with a much larger coverage than the ones mentioned in this report.

7.2 Orthogonality Constraint

As another direction of future work, we intend on continuing our exploration of how best to combine facets. In this report, concatenation emerged as the clear winner, even when the concatenated facets were projected down to a dimension of 300. It was discussed that one of the benefits of concatenation is that enforces orthogonality between facets, while summation does not. However, it is unclear how beneficial this orthogonality constraint is outside of the context of similarity/relatedness tasks and if this can explain downstream success of concatenation. As a way of testing this, we propose implementing *prismatic representations of words* (Kenyon-Dean, 2019), also known as *word prisms*.

The main innovation proposed in word prisms is that of *orthogonal summation*, which imposes orthogonality between word facets via orthogonal transformations. That is, for a word i, returning to the notation of Section 3, we would have:

$$|i\rangle = U_1|i\rangle_1 + U_2|i\rangle_2 + ... + U_n|i\rangle_n, \quad (5)$$

where the U_k, $k \in \{1, 2, ..., n\}$, are orthogonal matrices trained with the objective of enforcing orthogonality between each of the transformed facets for word i. With the imposition of this constraint, we can view the meta-embedding for a word i as the sum of linearly independent vector representations of the word i. Whether or not this can improve the performance of summation and be competitive with concatenation remains to be seen.

8 Conclusion

In this work, we apply existing methods for building meta-embeddings from distinct collections of word embeddings, namely the methods of summation and concatenation. In contrast to previous work on meta-embeddings, we focus on the combination of embeddings trained with different notions of context, rather than embeddings trained with different algorithms. We find that both summation and concatenation of these source embeddings (which we call "facets") leads to performance gains on five downstream tasks, most notably the task of supersense tagging. We also note that concatenation performs consistently better than summation, both on intrinsic and extrinsic evaluations. Projection of concatenated embeddings via principal component analysis proves to be a solid technique for dimensionality reduction which preserves most of the performance gains that the concatenation method provides.

In future work, we aim to integrate additional contexts into our meta-embeddings and to increase our vocabulary coverage. On top of this, we intend to explore whether or not the imposition of orthogonality between facets can compete with and even improve performance beyond the gains already shown in this paper.

9 Acknowledgements

This work would not have been possible without the tremendous support of Edward Newell, Kylie He, Kian Kenyon-Dean, and Jackie Chi Kit Cheung. Their input and feedback was invaluable.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Strakova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. pages 19–27.

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 278–289, Doha, Qatar. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Jeju Island, Korea. Association for Computational Linguistics.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada. Association for Computational Linguistics.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st workshop on evaluating vector-space representations for NLP*, pages 1–6.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602, Sydney, Australia. Association for Computational Linguistics.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. 2015. Sparse overcomplete word vector representations. *arXiv preprint arXiv:1506.02004*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. volume 20, pages 406–414.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Dave Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2007. English gigaword third edition. *Linguistic Data Consortium*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Kian Kenyon-Dean. 2019. Perspectives on prismatic representations of words.

Kian Kenyon-Dean, Andre Cianflone, Lucas Page-Caccia, Guillaume Rabusseau, Jackie Chi Kit Cheung, and Doina Precup. 2018. Clustering-oriented representation learning with attractive-repulsive loss. *CoRR*, abs/1812.07627.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. pages 142–150.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Edward Newell, Kian Kenyon-Dean, and Jackie Chi Kit Cheung. 2019. Deconstructing and reconstructing word embedding algorithms. *arXiv preprint arXiv:1911.13280*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *WWW*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Dongqiang Yang and David Powers. 2006. Verb similarity on the taxonomy of wordnet.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1351–1360.