Task-dependent Optimal Weight Combinations for Static Embeddings

Nathaniel R. Robinson, Carnegie Mellon University, USA nrrobins@cs.cmu.edu
Nathaniel Carlson, Brigham Young University, USA natec18@byu.edu
David R. Mortensen, Carnegie Mellon University, USA dmortens@cs.cmu.edu
Elizabeth Ann Vargas, Brigham Young University, USA elizag17@byu.edu
Thomas Fackrell, Brigham Young University, USA tfac1997@byu.edu
Nancy Fulda, Brigham Young University, USA nfulda@cs.byu.edu

Abstract A variety of NLP applications use word2vec skip-gram, GloVe, and fastText word embeddings. These models learn two sets of embedding vectors, but most practitioners use only one of them, or alternately an unweighted sum of both. This is the first study to systematically explore a range of linear combinations between the first and second embedding sets. We evaluate these combinations on a set of six NLP benchmarks including IR, POS-tagging, and sentence similarity. We show that the default embedding combinations are often suboptimal and demonstrate up to 12.5% improvements. Notably, GloVe's default unweighted sum is its least effective combination across tasks. We provide a theoretical basis for weighting one set of embeddings more than the other according to the algorithm and task. We apply our findings to improve accuracy in applications of cross-lingual alignment and navigational knowledge by up to 15.2%.

1 Introduction

Static word embeddings are used in a broad range of NLP applications, including conversational gameplay (Andrus and Fulda, 2020), text categorization (Minaee et al., 2021; Mitra et al., 2016), translation (Sabet et al., 2020; Jansen, 2017; Pourdamghani et al., 2018), affordance detection (Fulda et al., 2017a), and semantic analysis (Hamilton et al., 2016). In addition to using static embeddings directly, researchers often combine them with contextualized models or use them for embedding initialization of downstream tasks (Kocmi and Bojar, 2017) such as summarization (Lin et al., 2021) and neural machine translation (Qi et al., 2018). The persistence of static embeddings is due in part to their ease of use and low computational requirements. Rather than needing a forward pass through a neural network to embed each word, pre-trained embeddings can be stored in memory and retrieved with complexity $O(1)$.[1] §4.2 outlines more advantages of static embeddings.

[1]This is assuming a default Python or Java hash map. The worst case would be $O(n)$ with vocabulary size n, in the case of a trivially slow hash map, which is still well under transformer-based embedding retrieval complexity.

We propose an augmentation of three popular embedding methods (word2vec skip-gram, GloVe, and fastText). Word2vec skip-gram (Mikolov et al., 2013a) is a neural word context predictor, GloVe (Pennington et al., 2014) is a log-bilinear model that includes global context information with a co-occurrence matrix, and fastText (Bojanowski et al., 2017) incorporates sub-word information via character n-grams with a skip-gram objective to expedite training and handle unseen words. More details about these algorithms are in §2. Each of them produces two separate embedding sets ("target" and "context", see Figure 1) that we combine in previously unexplored ways. We show that these typically unexplored target and context combinations reveal much about embedding effectiveness. Our key contributions are as follows:

1. We provide a theoretical and empirical analysis of static embedding performance across weighted linear combinations of embedding sets ("target" and "context").
2. We generate 126 embedding sets from 6 corpora and show that the default target/context combination for each embedding algorithm is often sub-

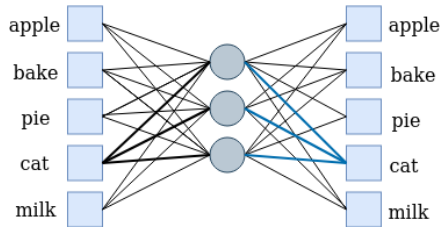

Figure 1: Illustration of "target" vectors (black lines, left) and "context" vectors (blue lines, right) produced by the word2vec skip-gram algorithm. Similar embedding pairs exist for fastText and GloVe. These are sometimes called "in" and "out" weights, respectively.

optimal.

3. We demonstrate improvements on analogies, textual similarity, IR, POS-tagging, cross-lingual alignment, and robotics navigation via embedding combinations and provide best practice recommendations.

We demonstrate up to 12.5% improvements over baseline performance on a diverse set of NLP benchmarks by combining target and context vectors. (See §5.) We analyze embeddings statistically and show that (1) word2vec target vectors encode better word-to-word relationships while context vectors are better suited for bag-of-words representations, (2) GloVe default vectors perform well on tasks for which they were tuned but under-perform generally, and (3) fastText target vectors' sub-word encodings are useful in many tasks but counterproductive for bag-of-words representations. (See §6.) Finally, in §7 we employ our methods practically to improve performance on MUSE (Lample et al., 2018) cross-lingual alignment by 0.69-1.56% and navigational robotics benchmarks by up to 15.2%

2 Background

We overview three common static embedding algorithms: word2vec skip-gram with negative sampling, fastText, and GloVe. Each of these algorithms has the same output: a set of embedding vectors, each corresponding to a word in a vocabulary. We outline applications that employ these different algorithms, but note that any task that uses vectors from one of these algorithms could just as feasibly use the vectors from another of them.

Mikolov et al.'s (2013a) **word2vec skip-gram** model learns embeddings as a neural regression problem: predicting each word's context. Each word in question is called the *target*, and its neighbors are called *context*. The model learns two sets of embeddings, corresponding to target and context words, though only the first is typically used. Word2vec is employed in many tasks, including measurement of MWE candidates (Pickard, 2020) and epidemic-related twitter stream classifications (Khatua et al., 2019). In our experiments we used skip-gram with negative sampling (rather than Mikolov et al.'s CBOW model) because of its comparability to GloVe and fastText. (See §3.) Throughout the text we refer to this algorithm simply as "word2vec."

The **fastText** algorithm (Bojanowski et al., 2017) integrates sub-word information into the skip-gram framework. It embeds character n-grams, and a word's embedding is the sum of its sub-word vectors. The context vectors are not composed of sub-words. Many applications use fastText, including hyperbolic word representations (Zhu et al., 2020) and low-resource sentence similarity (Khalid et al., 2021; Akhtar et al., 2017).

Popular **GloVe** embeddings (Pennington et al., 2014) are used for sarcasm detection (Khatri and P, 2020), emotion detection (Gupta et al., 2021), and lexical semantic analysis (Jain, 2020). GloVe's log-bilinear model learns two embeddings from word co-occurrence. By default, most public GloVe implementations sum the embeddings evenly. However, our study shows that this unweighted sum often does not maximize performance. (See §5.)

Our work challenges the assumption that the default combinations or selections of target and context vectors (*target only* for word2vec and fastText and *an unweighted sum* for GloVe) are optimal for any task, or even across tasks in general. We methodically explore a spectrum of target/context combinations for each algorithm and show that the default embedding selection is often not the best.

Although not directly studied here, other static embedding algorithms such as ConceptNet Numberbatch (Speer et al., 2017), hyperbolic word embeddings (Zhu et al., 2020), and word2vec-CBOW (Mikolov et al., 2013a) exist and merit study.

2.1 Contextual Embedding

The uses of static embeddings overlap with contextualized embedding models such as BERT (Devlin et al., 2019), ELMo (Peters et al., 2018), BART (Lewis et al., 2020), and others (Liu et al., 2019b; Robinson et al., 2021). These networks are adaptable to a variety of NLP tasks, from translation evaluation (Yuan et al., 2021; Zhang et al., 2019) to semantic tagging (Liu et al., 2019a). Some researchers have scrutinized them for consuming too many resources and lacking interpretability (Bender et al., 2021; Brown et al., 2020; Strubell et al., 2019). Static embeddings are employed instead in many practical NLP tasks because they are fast, computationally inexpensive, and intuitive.

Static embeddings are particularly suited to tasks that restrict predictions to a candidate set, such as word

analogies, since embeddings from these smaller models have a defined vocabulary that can be queried in nearest-neighbor search. Dufter et al. (2021) verified this trend for question answering and advocated for use of static embeddings because of their low computational cost: "'green' baselines are often ignored, but should be considered when evaluating resource-hungry deep learning models."

2.2 Prior Investigations of Context and Target Combinations

We are not the first to combine target and context embeddings. As mentioned in §2, GloVe implementations (Pennington et al., 2014) sum them by default. Nalisnick et al. (2016) used target vectors for queries and context vectors for documents in information retrieval (IR). They did not explore summing the two vector sets however.

Fulda and Robinson (2021) explored concatenating and summing word2vec skip-gram target and context vectors for analogies and sentence similarity. They found that with sufficiently large training corpora, target-context sums can outperform target embeddings (the default) and target-context concatenations. Their analysis reveals a theoretical advantage for summed embeddings in analogy tasks with dot-product-based similarity metrics: with target-context sums, the dot product between vectors for words a and b is $(t_a + c_a)^T(t_b + c_b) = t_a^T t_b + t_a^T c_b + c_a^T t_b + c_a^T c_b$, where t and c are target and context vectors. This is significant because, as Nalisnick et al. (2016) point out, the outer terms of this expression encode paradigmatic relations (e.g. *mason* and *carpenter*), while the inner terms encode syntagmatic relations (e.g. *mason* and *stone*). These different relations are relevant in analogies like *mason : stone :: carpenter : wood*.

Our work expands on and differs from these examples in four main ways: (1) we systematically explore *multiple* combinations of target and context vectors (e.g. 80% target added to 20% context, which we show in §6.1 may be conceptually closer to a true sum), (2) we apply these changes uniformly across word2vec, GloVe, and fastText on multiple training corpora and a variety of NLP tasks, (3) we provide a theoretical analysis of the properties of target and context vectors by task type, and (4) we identify a set of best practices and recommended embedding combinations for practitioners applying these algorithms in the wild.

3 Theoretical Motivation

This section shows that while the target and context vectors produced by the GloVe algorithm are fundamentally equivalent to each other, the same does not hold true in the cases of word2vec and fastText. This insight informs and motivates our analysis of various context/target embeddings in §6.

Word2vec Skip-gram Training The skip-gram model processes a corpus, considering each word as a target and its surrounding words within a window size w as context. For each step, the weights are updated for two objectives:

- **Objective 1**: Maximize the dot product between the target embedding for the target word and the context embeddings for its neighbors
- **Objective 2**: *Negative sampling*, or minimize the dot product between the target embedding for the target word and the context embeddings of random words

Theorem 1 shows that **objective 1** cannot account for any difference between target and context space. We outline terminology for the theorem below.

Terminology for Theorem 1: Let the operation $close(v_1, v_2)$ indicate an action: that of model weight updates to increase the dot product between vectors v_1 and v_2. All of the model's actions for **objective 1** as it processes the corpus once are

$$\{\{p(|j-i|)close(t_i, c_j)\}_{j=i-w, j\neq i}^{i+w}\}_{i=0}^{M-1} \quad (1)$$

where t_i and c_i are the target and context embedding for the word appearing in the ith position of the corpus, and M is the corpus length. The function $p(\ell)$ is a Bernoulli c.d.f. that returns 0 with probability $\frac{(\ell-1)}{w}$. This expresses how the skip-gram model probabilistically drops **objective 1** actions for context words far from the target. We require that $close(v_i, v_j)$ represent no action if either i or j is below zero or greater than $M-1$. Note that the arguments of $close$ may be commuted.

Given Theorem 1, all of the differences between target and context structure are due to **objective 2**. We show how common word context vectors behave like magnets for target vectors to cluster around, because of **objective 2** action distribution.

Let $far(v_1, v_2)$ indicate the action of weight updates to decrease the dot-product between v_1 and v_2 (the opposite of $close$). All **objective 2** actions from one reading of the corpus are

$$\{\{far(t_i, c_{x_j})\}_{j=0}^{5r-1}\}_{i=0}^{M-1} \quad (2)$$

where 5 is the set number of negative samples per word, r is a random integer $1 \leq r \leq w$ that depends on the output of p in Equation 1, and each c_{x_j} denotes the context vector for a word at index x_j drawn randomly from a distribution X of square-rooted word frequencies. Let X_{True} be the distribution of actual word frequencies from the corpus. When sampling from square-rooted word frequencies X, the most common words in

Theorem 1. ***Objective 1*** *does not cause any differences between target and context vector construction.*

Proof. Define $f(v_i, v_j) := p(|j-i|)close(v_i, v_j)$ for convenience, and note $f(v_i, v_j) = f(v_j, v_i)$. The set of total **objective 1** actions performed by the model as it reads the corpus once is then

$$\begin{aligned}
&\{\{f(t_i, c_j)\}_{j=i-w, j\neq i}^{i+w}\}_{i=0}^{M-1} \\
&= \{\{f(t_i, c_j)\}_{i-w\leq j\leq i-1}, \{f(t_i, c_j)\}_{i+1\leq j\leq i+w}\}_{i=0}^{M-1} \\
&= \{\{f(c_i, t_j)\}_{j-w\leq i\leq j-1}, \{f(c_i, t_j)\}_{j+1\leq i\leq j+w}\}_{j=0}^{M-1} \\
&= \{\{f(c_i, t_j)\}_{j-w\leq i\leq j-1}\}_{j=0}^{M-1} \cup \{\{f(c_i, t_j)\}_{j+1\leq i\leq j+w}\}_{j=0}^{M-1} \\
&= \{\{f(c_i, t_j)\}_{i+1\leq j\leq i+w}\}_{i=-w}^{M-2} \cup \{\{f(c_i, t_j)\}_{i-w\leq j\leq i-1}\}_{i=1}^{M-1+w} \\
&= \{\{f(c_i, t_j)\}_{i+1\leq j\leq i+w}\}_{i=0}^{M-1} \cup \{\{f(c_i, t_j)\}_{i-w\leq j\leq i-1}\}_{i=0}^{M-1} \\
&= \{\{f(c_i, t_j)\}_{i-w\leq j\leq i-1}\}_{i=0}^{M-1} \cup \{\{f(c_i, t_j)\}_{i+1\leq j\leq i+w}\}_{i=0}^{M-1} \\
&= \{\{f(c_i, t_j)\}_{j=i-w, j\neq i}^{i+w}\}_{i=0}^{M-1},
\end{aligned}$$

which is the original expression with the roles of t and c vectors reversed. Therefore c and t are reversible without changing the total **objective 1** actions performed by the model. I.e. their roles are identical in this process. □

the corpus will be less frequent, and the least common words will be more frequent, than when sampling from X_{True}. In Equation 1, we have $t_i, c_j \sim X_{\text{True}}$. However, in Equation 2, we have $t_i \sim X_{\text{True}}$ but $c_{x_j} \sim X$. This means that the context vectors c corresponding to frequent words will appear more often in the *close* function and less often in far (and context vectors for *infrequent* words will appear more often in far than in *close*). Therefore the target vectors will be biased to be more similar to the context vectors of high frequency words.

A brief analysis shows this. Let S_{100} be the 100 most common words in a corpus from the vocabulary V. For word2vec embeddings trained on three corpora, (Web Scraped, WikiReddit, and Wikipedia, described in §4.1) we calculated the cosine scores between all target vectors and the 100 context vectors for the words in S_{100}, $\text{scores}_1 = \{\cos(t_v, c_s) : v \in V, s \in S_{100}, v \neq s\}$ and the corresponding scores where target vectors came from S_{100}, $\text{scores}_2 = \{\cos(c_v, t_s) : v \in V, s \in S_{100}, v \neq s\}$. For each corpus, over 90% of the 100 highest scores in $\text{scores}_1 \cup \text{scores}_2$ were from scores_1, indicating that target vectors are clustered around common-word context vectors (more so than context vectors are clustered around common-word target vectors).

fastText vectors are trained using this same paradigm but with an additional difference between the embedding sets that is likely more important in applications: target vectors are composed by summing subword embeddings, while context vectors are not.

GloVe vectors are constructed differently than word2vec or fastText. Their training objective $(\sum_{i,j}^{V} f(X_{i,j})(t_i^T c_j + b_i + \tilde{b}_j - log(X_{i,j}))^2)$ is log-bilinear for target and context vectors. Thus there is no difference between target and context vector construction, short of random initialization. Analyses of vector space clustering and magnitude for GloVe, including the analysis described in the previous paragraph and statistics outlined in Table 4, reveal no notable differences between GloVe target and context distributions.

4 Methodology

We conduct experiments to answer three core questions: (1) How does task performance vary across linear combinations of target/context vectors, and do the default settings work generally well? (2) Is the pattern of performance (as a function of target/context weightings) similar in all three embedding algorithms? If not, what are the differences? (3) Are optimal weighting schemes data- and task-specific? If so, to what extent?

To answer these, we trained target and context embeddings for each of three algorithms (word2vec skip-gram, GloVe, fastText) on six corpora. We then produced more embeddings by combining each pair of target t and context c as follows:[2] $.8t + .2c$, $.6t + .4c$, $t + c$, $.4t + .6c$, and $.2t + .8c$. We refer to these combinations respectively as 80:20-sum, 60:40-sum, true sum, 40:60-sum, and 20:80-sum, in addition to target and context vectors alone. We generated 126 embedding sets total (six training corpora × three embedding algorithms × seven weighted sums). This spread of linear combinations has not been studied previously.

We insist that the target and context weights sum to 1 for the sake of uniformity and clarity in drawing

[2] $t + c$ is a stand-in for $.5t + .5c$, since our NLP tasks rely on magnitude-agnostic cosine similarity. We use $t + c$ to compare directly with existing methods.

conclusions from our experiments. Since our NLP applications employ magnitude-agnostic cosine distance as a similarity metric, allowing the weights to range from 0 to 1 is largely equivalent to letting them range from 0 to N for any positive real number N. We set $N = 1$ across all experiments to allow fair comparisons of weight values. It is worth noting that weights could also be negative, a possibility that is beyond the scope of our current study but that could be explored in future work.

Linear combinations are a form of summation. Summation is an established method for combining vector information, as discussed in §2.2. The fastText algorithm relies on sums of subword target vectors (Bojanowski et al., 2017). GloVe vectors are often target-context sums by default (Pennington et al., 2014). Target and context vectors ought to be reasonably well aligned for summation, since the objective of a static embedding model is to increase the dot products between target and context vectors, and the dot product is based on component-wise multiplication. This provides a theoretical motivation for our focus on target-and-context vector summation. Note, however, that summation is not the only possible way to combine target and context embeddings. As discussed in §2.2, Fulda and Robinson (2021) explored concatenating the embeddings instead. However, they found that summation yielded better results, a conclusion that they verified both theoretically and experimentally.

Despite the theoretical basis for their alignment, target and context vectors may not be perfectly aligned for summation in practice. One could employ a more intricate approach to align them, such as a linear mapping or an encoder/decoder reconstruction method such as MUSE (Lample et al., 2018). This is a potentially promising area of future research.

Note that our experimentation over a range of seven different target/context weighting values is comparable to performing a course grid search for this parameter in each task. Another more computationally expensive and task-specific method to tune this parameter is meta-learning. Our objective in our main experiments is to learn generalizable principles across a variety of tasks, which may be of value to other researchers, for which purpose we determined that this grid search approach would suffice. However, we do explore an application of optimizing the weighting via differentiation. See §7.2 for analysis.

Training hyperparameters We used embedding dimension 300. For **GloVe**: we used window size 10, minimum word count 5, and 25 training iterations. These have been shown to achieve optimal results (Pennington et al., 2014). The "minimum word count" mentioned is minimum frequency for a word in the corpus to include it in the model's vocabulary. For **fastText**: we used window size 10, minimum word count 5, 5 training epochs, and 3-6 character n-grams, as standard (Bojanowski et al., 2017). For **word2vec**: we used window size[3] 5 and 3 epochs, as recommended by Fulda and Robinson (2021). Due to the Web Scraped corpus' size and computing restraints, we opted for minimum word count 100.

4.1 Training Corpora

Our six training corpora are in Table 1. The Web Scraped corpus was generated to imitate the unreleased WebText data used to train OpenAI's GPT-2 (Peterson, 2019; Radford et al., 2019). The Wikipedia corpus is a collection of all text on Wikipedia from 2004. The WikiReddit dataset is the concatenation of the Wikipedia corpus with text from Reddit. The Toronto Books Corpus contains 11,038 books collected by the University of Toronto (Zhu et al., 2015). The smallest corpus consists of classic books from Project Gutenberg (Lahiri, 2014). Spanish Wikipedia was the dump from October 20, 2021 collected using the WikiExtractor tool (Attardi, 2015).

We had the majority of these corpora on hand and used them to reduce computational expense. Though the NER task could potentially have benefited from using newer corpora, such as a more recent download of English Wikipedia, we did not see a clear theoretical impact from using newer corpora on the other evaluation tasks. We provide a brief analysis of the interaction between embedding performance and corpus choice in 6.4. An in-depth analysis of this interaction is outside the scope of this paper, but we recommend it as an area of further study.

In Table 2 we show the vocabulary sizes for the embedding sets trained using the three embedding algorithms and the six training corpora in our experiment set.

Corpus	Size	Tokens
Web Scraped	59.0 GB	9.6B
WikiReddit text	21.0 GB	4.1B
Wikipedia text	16.7 GB	2.8B
Toronto Books	4.6 GB	984M
Classic Books	20.3 MB	<1M
Spanish Wikipedia	4.5 GB	667M

Table 1: Training corpora (five English, one Spanish). Novel corpora will be released upon acceptance.

[3]Our word2vec implementation denotes unidirectional window size. This value is equivalent to the bi-directional window size 10 for fastText and GloVe.

	word2vec	fastText	GloVe
Web Scraped	408K	3.36M	3.36M
WikiReddit text	530K	2.51M	2.51M
Wikipedia text	432K	1.95M	1.95M
Toronto Books	92.3K	315K	315K
Classic Books	5.69K	36K	36K
Spanish Wikipedia	347K	2.24M	2.24M

Table 2: Vocabulary sizes for each embedding set trained on each of the corpora, using the hyperparameters delineated in §4 (using 3 significant figures)

4.2 Evaluation Tasks

We evaluated each embedding set on six NLP tasks chosen to represent a broad sampling of static embedding uses, conducting over 650 evaluations. We describe task details below for reproduction.

We employed two analogy question evaluations. **The Google Analogy Test Set** (Mikolov et al., 2013c,b) is a common embedding evaluation benchmark. It contains 19,544 analogy questions in 14 categories: six semantic (e.g. family relationships, countries and capitals) and eight grammatical (e.g. adjectives with superlatives). GloVe vectors were tuned for this benchmark. **Turney's (2006) set of SAT questions** was used by Fulda and Robinson (2021) to evaluate static embeddings. It contains 374 analogy questions with semantic relationships like the mason/carpenter example in §2.2.

Selection of analogy candidates: Given analogy A : B :: C : D, in the Google test, embeddings predict D given A, B, and C from $\hat{d} = \arg\max_{d' \in S} \cos(b - a + c, d')$, where a, b, and c are vectors corresponding to their respective words, and S is the set of all vectors in the embedding. *cos* denotes cosine similarity. In the SAT test, embeddings predict C and D given A and B from $\hat{c}, \hat{d} = \arg\max_{(c',d') \in S} \cos(b - a, d' - c')$, where S is a set of four multiple-choice candidate pairs (c_i, d_i). To illustrate more intuitively, say we have the example analogy *mason : stone :: carpenter : wood*. In the Google test paradigm, our task is to predict "wood" given "mason," "stone," and "carpenter." And we do this by finding the word in our vocabulary whose vector is closet to $v_{stone} - v_{mason} + v_{carpenter}$. In the SAT paradigm, we are given the pair "mason" and "stone," along with a list of multiple-choice options, each containing a pair of words and one of which contains the pair "carpenter" and "wood." We select the option whose pair-wise difference vector is closest to the difference vector for the given pair. Because of the high propensity of esoteric words in SAT analogies, we skip SAT questions with out-of-vocabulary words.

SemEval 2013 (Wilson et al., 2013) is a sentence textual similarity (STS) set of 1,379 sentence pairs with human-given similarity scores. We sum word vectors to obtain sentence vectors, then measure how well pairwise cosine similarity correlates with gold similarity using Spearman's rho.

We adapt Nalisnick et al.'s (2016) **IR method**, the Dual Embedding Space Model (DESM). We collected 36,701 queries and 3.2 million documents from Ni (2015). Each query is mapped to a list of 100 relevant documents with relevance scores. For query-document similarity we calculate a modified DESM score $\text{DESM}(Q, D) = \frac{1}{|Q|} \sum_{q \in Q} \frac{q^T \overline{D}}{||q|| ||\overline{D}||}$, where Q and D are matrices of vectors for the query and document words,[4] respectively, and $\overline{D} = \frac{1}{|D|} \sum_{d \in D} d$. We rate embeddings by average Spearman's Rho correlation between DESM scores and ground truth document relevance scores across queries.

Our method for **POS tagging** is adopted from Premjith (2019). We predict eight POS categories (noun, adjective, adverb, adposition, determiner, pronoun), using the highest-performing of five classifiers: K-neighbors, decision tree, random forest, multi-layer perceptron, and Gaussian naive Bayes. These are the models and parameters used by Premjith (2019). In recent years, other models such as LSTM and encoder/decoder models have been commonly used for POS-tagging. We opted to keep the NLP applications we gathered as close to their original form as possible, for the sake of uniformity, and as such we do not augment the model list with these additional architectures. We acknowledge that a POS-tagging embedding evaluation employing LSTM and seq2seq models could be worthwhile in future studies. In all cases, embedding vectors were used as input to the tagging models. We evaluate both classifiers and embeddings with POS prediction weighted-averaged F1 score.

Our **cross-lingual alignment task** is adapted from Jansen (2017). We train a transition matrix between English and Spanish Wikipedia embeddings on a 2894-word English-Spanish dictionary with a 64-16-20 train-dev-eval split. We train for 10 epochs with learning rate .001 and the Adam optimizer (Kingma and Ba, 2015). We rate embeddings by validation accuracy. In each test, the settings of Spanish and English embedding training are identical (same embedding algorithm and target/context combination).

Tasks summary: This set of evaluations contains both common embedding benchmarks and practically relevant tasks. Static embeddings are particularly suited to multiple of these tasks because they involve context-less word relationships and proximity searches across stored embedding sets.

[4] Excluding stop words such as "and," "that," or "may", as defined by SpaCy model `en_core_web_sm`. See https://spacy.io/models/en.

5 Results

Our results show that the default settings of each algorithm for combining target and context vectors do not always perform best, and often perform worst, on NLP tasks. Figure 2 shows the performance of 7 target/context combinations across the 5 English corpora and 3 embedding algorithms.

In summary, for **word2vec**, context vectors perform best on IR, but target vectors are best on SAT and Google analogy tasks and cross-lingual alignment, and summed vectors excel in STS. **GloVe** vectors exhibit two major trends. Summed vectors perform best in STS but worst in other tasks: SAT analogies, POS tagging, and IR. **fastText** target vectors perform best on Google analogies and POS tagging, but in IR and cross-lingual alignment, summed vectors excel.

Table 3 shows the percentage improvement from tuning the target/context weighting over default weighting for each embedding algorithm and evaluation task. These values represent the improvement from the highest performance using the default target/context setting to the highest performance after our search over linear combinations. We see the largest performance increase, 12.5%, in the case of word2vec on the STS task.

Patterns across target and context combinations are dependent on the NLP task and algorithm. This suggests that tuning the target/context summation weight (rather than using defaults) can improve performance markedly. For example, GloVe's default true sum does well on the STS task but under-performs on SAT analogy, POS, and IR. Word2vec and fastText's default target embeddings perform well on the Google Analogy and POS tasks but under-perform on the IR task.

Performance trends are generally consistent across our five training corpora. There are some notable exceptions to this generality, which we discuss in §6.4.

Average Effect of Target/Context Weighting Across Corpora and Tasks Figure 3 shows the average performance of each algorithm across all corpora and tasks. Results suggest fastText performs optimally with a 20:80 target/context combination rather than the default setting of 100% target. GloVe performance is highest at 80:20 and 20:80. These results suggest that a 20:80 or 80:20 combination of target and context may be an advantageous default for future embedding sets, especially in settings where hyperparameter tuning is not possible. (E.g. because embeddings are pre-trained or due to computational constraints.) In §6 we analyze results, and in §7 we present practical applications for these observations.

6 Analysis

This section provides theoretical backing for the observed performance of target vs. context vectors on specific types of tasks. We analyze the advantages of using weighted target and context combinations for specific use cases and offer recommendations for best practices in static embedding research.

6.1 Word2vec Analysis

Word2vec target vectors outperform their context counterparts in analogy tasks, implying that the phenomena described in §3 encode stronger word relationships in target space. We show how word2vec target vectors are advantaged in word-search style tasks like analogies and cross-lingual alignment, while context vectors have the advantage in document-level tasks like STS and IR.

Because context vectors do not attract and repel target vectors with equal frequency (see §3), there is higher variability in their length; vectors likely become large to produce high positive dot-products with neighbors or low negative dot-products with non-neighbors. Figure 4 shows that the norms of context vectors are an order of magnitude larger than those of target vectors. This justifies the use of weighted sums. An unweighted addition of small target and large context vectors results in a set that resembles context space. An 80:20-sum may be closer to the ideal of an even sum.

Further analysis suggests that context vectors are ill-equipped for some semantic tasks. Table 4 shows that context space has higher inertia in k-means clustering, indicating that it is harder to cluster into meaningful semantic groups.

Further statistics gathered from word2vec target and context spaces are surprising. (See Table 4.) The extremely small mean, minimum, and maximum cosine distances from the centroid vector and the small standard deviation imply that context vectors are clustered tightly in cosine distance around a centroid. The high skewness and extremely high kurtosis indicate existence of extreme outliers. These properties increase the likelihood of selecting an incorrect vector in tasks that search the vocabulary space, such as analogies and cross-lingual alignment (where context vectors perform worst). Target vectors show none of these disadvantages.

In contrast, although context vectors perform worse on these word-search tasks, they are well-suited to tasks in which word vectors within a sentence or document are summed to form bag-of-words representations, such as STS or IR. Context vectors' more variable norms play to their advantage here, making them preferable to target vectors. Table 5 shows how context vectors for stop words (defined by SpaCy model

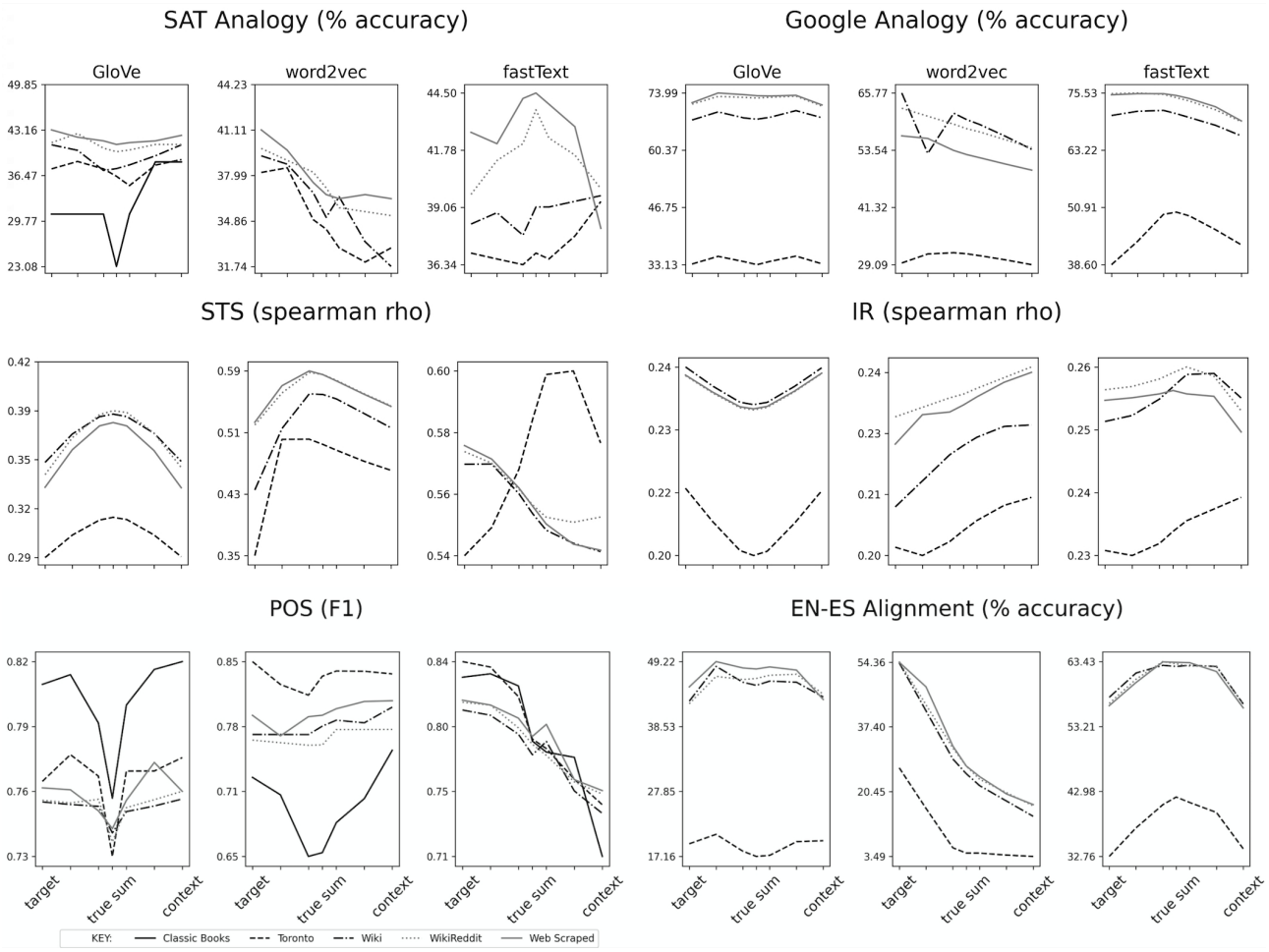

Figure 2: Algorithm performance grouped by task. X-axis ticks correspond left to right to (target, context) weightings of (1.0,0.0), (0.8,0.2), (0.6,0.4), (0.5,0.5), (0.4,0.6), (0.2,0.8), and (0.0,1.0). Results for the smallest corpus are omitted when they are so poor that they impair the visible contrast between other scores.

	SAT Ana.	Google Ana.	STS	IR	POS	EN-ES
word2vec	0%	0%	12.5%	5.27%	0%	0%
fastText	4.39%	0.2%	3.90%	1.52%	0%	9.72%
GloVe	5.22%	0.9%	0%	3.00%	8.8%	2.52%

Table 3: Percentage improvement by target/context weight tuning, over default target/context weighting, for each embedding algorithm and task shown in Figure 2. Improvement of 0% indicates that the default weighting performed best.

`en_core_web_sm`) are smaller than average. This equips context vectors for tasks involving sums of word vectors. It means that vectors carrying less semantic information will be play a less significant role in bag-of-words sentence representations, which will then be less noisy and more closely resemble the vectors for their meaningful keywords.

6.2 fastText Analysis

The fastText algorithm constructs vectors in a similar way to word2vec. However, fastText vectors display different performance patterns from word2vec across tasks. Recall from §3 that fastText target vectors (and not context) benefit from sub-word information. This seems to play a large role in their performance. Sub-word information is useful in POS-tagging, where English morphology can indicate part of speech, and Google analogies, which involve both derivational and

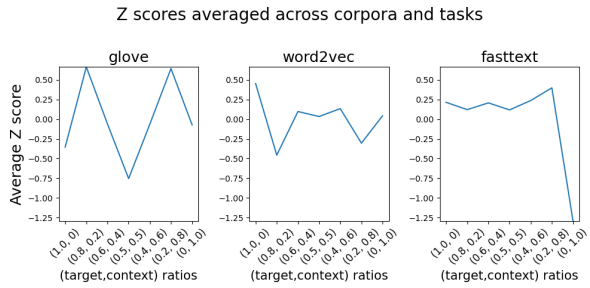

Figure 3: Average Z-scores (standard deviations from the mean) across all training corpora and all tasks in Figure 2.

	w2v tgt	w2v ctx	GloVe tgt	GloVe sum	fT tgt	fT ctx
mean	0.74	0.03	0.76	0.73	0.53	0.29
std. dev.	0.18	0.03	0.20	0.23	0.06	0.13
skewness	-0.04	7.63	-0.18	-0.12	0.22	0.75
kurtosis	-0.31	1e+2	0.06	-0.18	0.15	0.61
min.	0.16	3e-3	0.06	0.04	0.21	2e-3
max.	1.36	0.96	1.90	1.89	0.87	1.26
mode.	1.15	0.46	1.50	1.67	0.56	1.10
inertia (1e6)	1.13	5.31	3.73	11.0	24.8	28.3

Table 4: **Statistics on cosine distances from each vector to centroid** for embeddings trained on Wikipedia, and average inertia for k-means clustering over 6 values of k ($k \in \{5, 10, 15, 20, 25, 30\}$).

	w2v tgt	w2v ctx
Stop word avg. norm	2.70	7.07
Content word avg. norm	2.22	19.66
Stop norm as % of content norm	122%	36.0%

Table 5: In context space (and not target), stop words have smaller norms than other words.

inflectional morphological processes. It also appears useful for semantic textual similarity whenever large training corpora are used.

Interestingly, context vectors perform best in the IR test, which involves summing the word vectors in long documents to form (bag-of-words) document representations $\bar{D}$. Since fastText target vectors are already bag-of-subwords representations (as noted in §2), the treatment of documents as large bags of unordered subwords may dilute the usefulness of the representation.

A particularly notable trend is performance on the cross-lingual alignment task. As expected from its widespread application to multilingual settings, fastText outperforms GloVe and word2vec. But in contrast to the conventional use of target vectors, target-context sums are the best-performing combinations. (In §7.1 we apply these vectors to a similar application and find 80:20 sums to be the best combination.) We hypothesize that this is because of morphological differences between Spanish and English. Sub-word information is useful for interpreting meaning in both languages, but over-dependence on these characteristics may cause failures due to distributional differences in morphology. Analysis of these trends across more language pairs is a topic for future research.

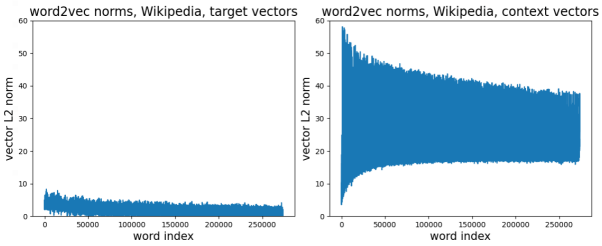

Figure 4: Norms of word2vec target and context vectors trained on Wikipedia, ordered from most common tokens to least

6.3 GloVe Analysis

As discussed in §3, GloVe target and context space are structurally similar. As a result, GloVe performance graphs in Figures 2 and 3 are mostly symmetrical. This leaves the question of why sums perform well on some tasks and poorly on others.

Recall from §4.2 that the Google Analogy Test is composed of nine sub-tests of grammatical analogies and five of semantic analogies. We analyzed vector performance on a fine-grained breakdown of Google Analogy subsets and found that in individual sub-tests, performance varies in a regular way: sums perform well on semantic analogies and poorly on grammatical analogies. Results from two sub-tests are in Figure 5. It appears GloVe's true sum vectors (its default) configured to these semantic questions when the algorithm was tuned on Google analogies, perhaps because the two largest sub-tests in the test set are capital-country and city-state relations. Observe the high accuracy achieved by GloVe default on the semantic task in Figure 5 (90–98%). Trends suggest, however, that the default sum performs worse more generally: In grammatical Google analogy sub-tests, SAT analogies, IR, and POS. Analysis backs this finding: True sum space has higher inertia in k-means clustering than target space (see Table 4), suggesting it is more difficult to cluster meaningfully, and it is the least robust GloVe combination (see Figure 3.)

6.4 Corpus Effect

Performance trends in our experiments are generally consistent across training corpora. Because of the few

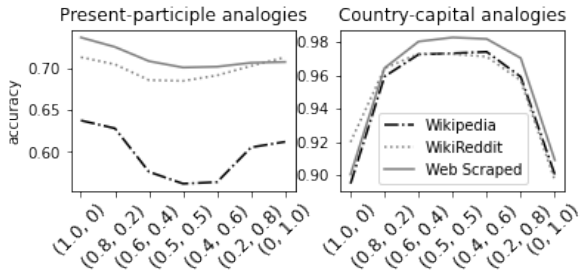

Figure 5: Example GloVe results across embedding combinations for grammatical (left) and semantic (right) analogies

exceptions to this generality and our primary focus on target/context weighting, we did not conduct an in-depth analysis of their causes. Here we note some observations briefly.

For fastText on SAT Analogies, Web Scraped and WikiReddit exhibit different trends from the other corpora. As Table 2 shows, these two corpora yield larger vocabularies than the others. Since SAT analogies rely on embeddings for less frequent vocabulary words, this could be a cause for the trend difference.

We see an aberration from general trends in the performance of fastText vectors trained with Toronto Books on STS. The STS test consists of many pairs of photo captions. The Toronto Books Corpus consists of 61.2% fiction books (Bandy and Vincent, 2021), which is likely rich in descriptive language. The STS test suite also contains answer comparisons and news captions. Dialogue in fiction books could be beneficial for detecting answer similarity. And non-fiction books may be particularly helpful for news caption comparisons. It is difficult to discern why these benefits would be manifest in the case of fastText and with an aberrant target/context trend. Drawing more conclusions may require a more in-depth study into corpus effect on these summed embeddings.

Note that in some experiments, the weighting of target and context determines whether performance from the Classic Books corpus can be comparable to the other corpora. This suggests an opportunity for target/context weight tuning to improve results in very low-resource settings. We discuss this more in §7.4.

7 Applications

Based on our findings, we recommend that practitioners tune the target/context embedding weight for downstream applications. The following two examples demonstrate that by doing so, it is possible to improve upon default vector performance using applications from published literature.

Note that our baselines differ from published results, which employed downloadable pre-trained embeddings that were extensively optimized. This approach yields high-quality results, but it does not allow for comparison of target/context combinations since only one combination is typically released online. We therefore used our own embeddings, trained on the Wikipedia corpus, to explore relative performance across a spectrum of target/context combinations. We encourage future practitioners to release both embedding sets.

7.1 MUSE Cross-lingual Alignment

Cross lingual alignment is a popular approach to multilingual text representations. Algorithms learn a transformation matrix to map embeddings from one language into another. Lample et al. (2018) accomplish this via an unsupervised adversarial algorithm, MUSE.

Using Lample et al.'s methods, we present results for different target/context combinations of fastText vectors. We selected fastText for this experiment to match Lample et al.'s (2018) implementation, and additionally because fastText performed the best on supervised cross-lingual alignment in §5 (the true sum combination, to be precise). To score in this unsupervised case, we query a fixed number of source word embeddings and measure accuracy for correct target retrieval for $k = 1, 5, 10$ nearest neighbors. In this task, we found that the 80:20 sum outperformed all other combinations. See Table 6. Interestingly 20:80 sum and context vectors perform significantly worse than the other combinations we tested, suggesting that the absence of the sub-word enriched target embeddings leads to degradation of performance.

7.2 Harvesting Common-sense Navigational Knowledge for Robotics

In this section we present an additional recommendation to practitioners. When optimizing for target/context weight via differentiation, take care with the choice of objective function, and consider the complexity of high-dimensional vector spaces.

Fulda et al. (2017b) used a novel distance metric to extract navigational relationships between objects for robotics applications, the Directional Scoring Method (DSM). They evaluated this method on a series of ground-truth object relations, contained in the BYU Analogical Reasoning Dataset. (See https://github.com/NancyFulda/BYU-Analogical-Reasoning-Dataset.) Using word2vec skip-gram vectors as the original authors did, we ventured to find the optimal target/context weight $\lambda \in [0, 1]$ for this application (where for each word w, the embedding vector $v_w = \lambda t_w + (\lambda - 1)c_w$). We grid

	default	*80:20 sum*	*60:40 sum*	*50:50 sum*	*40:60 sum*	*20:80 sum*	*context*
k=1	64.14%	**65.70%**	65.55%	63.69%	62.49%	58.02%	51.34%
k=5	81.02%	**81.83%**	81.75%	80.52%	79.56%	75.94%	69.56%
k=10	84.91%	**85.60%**	85.50%	84.57%	83.86%	80.72%	75.25%

Table 6: MUSE alignment accuracy percentages with fastText English and Spanish vectors. Best results are **bold**. The 80:20 combination outperforms all others.

searched over eleven λ values to maximize accuracy ($\lambda \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$). The default target vectors were quite effective for this application. $\lambda = 1.0$ scored highest for 6 of the 11 analogy categories (accessing containers, affordance, causation, containers, locations for objects, rooms for containers). Results for the other 5 categories are in Table 7.

	belong	rooms objects	tools	trash treas.	travel
default	6.21%	21.1%	3.57%	37.0%	17.6%
λ *GS*	9.66%	21.5%	6.90%	52.2%	18.4%
$\lambda =$	0.9	0.8	0.8	0.4	0.9

Table 7: DSM scores where λ grid search (GS) improved performance. *Default* indicates $\lambda = 1$.

We found that optimizing λ via differentiation was not effective. We constructed an objective, the sum of directional scores for each of the ground-truth analogy answers in a training set, 60% of analogical questions, and maximized the objective over λ. Because this objective minimized DSM-distance between ground truth vectors and DSM-hypothesis vectors, the optimal value of λ defaulted to the most tightly clustered vector space available. In our case with word2vec, $\lambda = 0$ was selected every time, since word2vec context vectors are tightly clustered in DSM space. This did not maximize the probability of choosing the correct vector.

Modifying the objective function by subtracting directional scores for all incorrect answers circumvented this issue and allowed for more diverse selections of optimal λ, but it still was not effective. We found that the value of λ that maximizes DSM scores for all correct answers and minimizes scores for all incorrect answers does not necessarily result in higher answer accuracy, regardless of whether we used a subset of questions for training or the whole set. This is likely due to intractable subtleties in high-dimensional vector geometries, a phenomenon for further investigation.

The results from this application do not preclude the use of differentiation or other meta-learning techniques to tune the target/context weight λ for other tasks. We strongly encourage such investigations. But until further conclusions, we urge that objectives to optimize λ be attempted with caution and thorough verification.

7.3 Summary Recommendations

Our observations suggest that values in Table 8 may be the most reasonable choices of target/context combinations for each embedding algorithm and task. Because even similar tasks may differ in nature, however, we encourage all practitioners to optimize target/context weights via grid search whenever possible.

While our recommendation is to tune/optimize, we recognize that many researchers, especially those applying word embeddings in research areas outside of computational linguistics, may not have the resources to tune their own weight parameters. We therefore provide recommendations on which simple linear combinations are broadly applicable to various task types.

7.4 Potential for Low-resource and Multilingual NLP

Our results suggest that tuning target/context weight can, in some cases, elevate the performance of low-resource embeddings to the level of higher-resource systems. One of the most promising areas for improvement of static word representations is NLP for low-resource languages. Many low-resource languages do not have high-quality contextual embedding tools such as BERT (Devlin et al., 2019) and lack the resources to train data-hungry BERT-like models. Many of these languages rely on improvements in static embedding technologies for accurate representations in NLP.

Multiple of the word2vec, GloVe, and fastText applications listed in §1 and §2 are for low-resource domains. Among the works referenced in this paper alone, we find examples of static embeddings applied to technologies for Amharic, Azerbaijani, Belarusian, Bengali, Galician, Gujarati, Hausa, Marathi, Punjabi, Somali, Tamil, Telugu, Uighur, Urdu, Uzbek, Yoruba, and more (Qi et al., 2018; Pourdamghani et al., 2018; Khalid et al., 2021; Akhtar et al., 2017).

A major limitation of our study is that, given the large number of independent variables we tested already, we were constrained to applications involving English (and some with Spanish, another high-resource language). Unfortunately, this limitation inhibits us

	word2vec	fastText	GloVe
word search tasks	target	**target-heavy/true sums**	target/context
sentence textual similarity	**true sum**	target/20:80 sum	true sum
bag-of-words representations	context	**context-heavy sums**	target/context
semantic Google analogies	target	target-heavy/true sums	**true sum**
grammatical analogies	target	**target-heavy/true sums**	target/context
other analogies	target	**target-heavy/true sums**	target/context
cross-lingual alignment	target	**80:20 sum/true sum**	80:20 sum
POS tagging	**target/context**	target	target-/context-heavy
overall	target	**target/20:80 sum**	80:20 sum/20:80 sum

Table 8: Recommended target/context embedding usage by task and embedding algorithm. The algorithm that performed best on each task type in our experiments is **bold**. Practitioners are cautioned that even similar tasks may differ in nature, and that the general trends indicated here may not hold in all use cases.

from drawing concrete conclusions about performance trends and their dependence on training language. This is a primary area of potential for future research. We hope to see more targeted studies addressing the effectiveness of target/context weight tuning on low-resource tasks, particularly for fastText vectors, which are often used in multilingual settings. Since fastText vectors formed by target-context sums combine morpheme information with full word information, they could be valuable in applications for morphologically rich languages, such as Arabic, Finnish, and Quechua.

8 Conclusion

By leveraging unconventional combinations of target and context vectors learned by GloVe, fastTest, and word2vec, we achieve improvements of up to 12.5% on common word embedding tasks such as POS-tagging and IR, thus elevating the usefulness of these popular and inexpensive word representations for NLP tasks. Experiments with 126 embedding sets on six generic tasks and two downstream applications show that tuning the hyperparameter of target and context weight for downstream tasks can improve performance significantly over default settings, increasing accuracy by 0.69% to 1.56% on MUSE cross-lingual alignment and by up to 15.2% on navigational robotics benchmarks.

Analysis suggests that target-heavy word2vec combinations are most suited to tasks involving single-word relationships, while context vector information is useful in summed sentence representations. We further observe that GloVe default settings perform best on tasks for which GloVe was tuned but tend to perform poorly on others, and that fastText target vectors excel in tasks such as POS-tagging, where sub-word information is particularly relevant. These findings reveal a disconnect between the maximum potential of static embedding algorithms and the ways in which they are typically used. In a majority of cases, the performance of pre-trained word embeddings could be improved by tuning the target/context weight hyperparameter. Furthermore, because a target/context weighting is typically chosen prior to the release of extensively pre-trained word vectors, the possibility of exploring various target/context weightings has typically not been made available to subsequent researchers. In alignment with our results, we urge those who design and train static word embedding models to release both target and context vector sets.

The software and embeddings used in our experiments will be released publicly under the MIT license. Given the widespread use of GloVe, word2vec, fastText, and other static embeddings, there is a need for deeper understanding of target and context interactions. Directions of future work in this area include the semantic content contained in context and target embeddings; the interplay between embedding algorithm, corpus size, and corpus genre; vector normalization methods to avoid norm imbalance; distance metrics not based on cosine similarity; and paired-embedding algorithms where context and target spaces are used individually.

References

Akhtar, Syed Sarfaraz, Arihant Gupta, Avijit Vajpayee, Arjit Srivastava, and Manish Shrivastava. 2017. Word similarity datasets for Indian languages: Annotation and baseline systems. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 91–94, Valencia, Spain. Association for Computational Linguistics.

Andrus, Berkeley and Nancy Fulda. 2020. Immersive gameplay via improved natural language understanding. In *Foundations of Digital Games 2020*.

Attardi, Giusepppe. 2015. Wikiextractor. https://github.com/attardi/wikiextractor.

Bandy, John and Nicholas Vincent. 2021. Addressing "documentation debt" in machine learning: A retrospective datasheet for bookcorpus. In *Proceedings*

of the Neural Information Processing Systems Track on Datasets and Benchmarks, volume 1.

Bender, Emily M., Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

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

Brown, Zachary, Nathaniel Robinson, David Wingate, and Nancy Fulda. 2020. Towards neural programming interfaces. *Advances in Neural Information Processing Systems*, 33:17416–17428.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dufter, Philipp, Nora Kassner, and Hinrich Schütze. 2021. Static embeddings as efficient knowledge bases? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2353–2363.

Fulda, Nancy, Daniel Ricks, Ben Murdoch, and David Wingate. 2017a. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1039–1045.

Fulda, Nancy and Nathaniel Robinson. 2021. Improved word representations via summed target and context embeddings. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*.

Fulda, Nancy, Nathan Tibbetts, Zachary Brown, and David Wingate. 2017b. Harvesting common-sense navigational knowledge for robotics from uncurated text corpora. In *Proceedings of the First Conference on Robot Learning*.

Gupta, Piyush, Inika Roy, Gunnika Batra, and Arun Kumar Dubey. 2021. Decoding emotions in text using glove embeddings. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 36–40.

Hamilton, William L., Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. *CoRR*, abs/1605.09096.

Jain, Vaibhav. 2020. GloVeInit at SemEval-2020 task 1: Using GloVe vector initialization for unsupervised lexical semantic change detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 208–213, Barcelona (online). International Committee for Computational Linguistics.

Jansen, Stefan. 2017. Word and phrase translation with word2vec. *CoRR*, abs/1705.03127.

Khalid, Usama, Aizaz Hussain, Muhammad Umair Arshad, Waseem Shahzad, and Mirza Omer Beg. 2021. Co-occurrences using fasttext embeddings for word similarity tasks in urdu. *CoRR*, abs/2102.10957.

Khatri, Akshay and Pranav P. 2020. Sarcasm detection in tweets with BERT and GloVe embeddings. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 56–60, Online. Association for Computational Linguistics.

Khatua, Aparup, Apalak Khatua, and Erik Cambria. 2019. A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks. *Information Processing & Management*, 56(1):247–257.

Kingma, Diederik P and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Kocmi, Tom and Ondřej Bojar. 2017. An exploration of word embedding initialization in deep-learning tasks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, pages 56–64, Kolkata, India. NLP Association of India.

Lahiri, Shibamouli. 2014. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden. Association for Computational Linguistics.

Lample, Guillaume, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *International Conference on Learning Representations*.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,

pages 7871–7880, Online. Association for Computational Linguistics.

Lin, Zhou, Qifeng Zhou, and Langcai Cao. 2021. Two-stage encoder for pointer-generator network with pretrained embeddings. In *2021 16th International Conference on Computer Science & Education (ICCSE)*, pages 524–529. IEEE.

Liu, Nelson F., Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *NIPS*, pages 3111–3119. Curran Associates, Inc.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Minaee, Shervin, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3).

Mitra, Bhaskar, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. This paper is an extended evaluation and analysis of the model proposed in a poster to appear in WWW'16, April 11 - 15, 2016, Montreal, Canada.

Nalisnick, Eric, Mitra Bhaskar, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *WWW '16 Companion: Proceedings of the 25th International Conference Companion on World Wide Web*, pages 83–84.

Ni, Chien-Chun. 2015. Multiple choice question (MCQ) dataset. https://www3.cs.stonybrook.edu/ chni/post/mcq-dataset/.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peterson, Joshua C. 2019. OpenWebText. https://github.com/jcpeterson/openwebtext.

Pickard, Thomas. 2020. Comparing word2vec and GloVe for automatic measurement of MWE compositionality. In *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pages 95–100, online. Association for Computational Linguistics.

Pourdamghani, Nima, Marjan Ghazvininejad, and Kevin Knight. 2018. Using word vectors to improve word alignments for low resource machine translation. 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 524–528, New Orleans, Louisiana. Association for Computational Linguistics.

Premjith, B. 2019. Part of speech tagging machine learning deep learning word2vec fastText. https://github.com/premjithb/Part-of-Speech-Tagging-Machine-Learning-Deep-Learning-Word2vec-fasttext.

Qi, Ye, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? 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

529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Robinson, Nathaniel, Zachary Brown, Timothy Sitze, and Nancy Fulda. 2021. Text classifications learned from language model hidden layers. In *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000207–000210. IEEE.

Sabet, Masoud Jalili, Philipp Dufter, and Hinrich Schütze. 2020. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *CoRR*, abs/2004.08728.

Speer, Robyn, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Strubell, Emma, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Turney, Peter D. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Wilson, Theresa, Zornitsa Kozareva, Preslav Nakov, Alan Ritter, Sara Rosenthal, and Veselin Stoyanov. 2013. Sentiment analysis in twitter. http://www.cs.york.ac.uk/semeval-2013/task2/.

Yuan, Weizhe, Graham Neubig, and Pengfei Liu. 2021. BARTScore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Zhang, Tianyi, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

Zhu, Yudong, Di Zhou, Jinghui Xiao, Xin Jiang, Xiao Chen, and Qun Liu. 2020. HyperText: Endowing FastText with hyperbolic geometry. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1166–1171, Online. Association for Computational Linguistics.

Zhu, Yukun, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724.