Learning Similarity-Preserving Meta-Embedding for Text Mining

Jidapa Thadajarassiri
Data Science Program
Worcester Polytechnic Institute
Worcester, USA
jthadajarassiri@wpi.edu

Cansu Sen
Computer Science Department
Worcester Polytechnic Institute
Worcester, USA
csen@wpi.edu

Thomas Hartvigsen
Data Science Program
Worcester Polytechnic Institute
Worcester, USA
twhartvigsen@wpi.edu

Xiangnan Kong
Computer Science Department
Worcester Polytechnic Institute
Worcester, USA
xkong@wpi.edu

Elke Rundensteiner
Computer Science Department
Worcester Polytechnic Institute
Worcester, USA
rundenst@wpi.edu

***Abstract*—Publicly available pre-trained word embeddings are rich sources for turning critical high-dimensional representations of huge text data repositories into meaningful compact vectors essential for text mining applications. With many of such pre-trained embedding sources available, each faces limitations in the appropriateness of their language use for the downstream text-mining tasks. *Meta-embeddings* aim to tackle this ambiguity challenge by fusing multiple embedding sources into one feature space. However, current *meta-embedding* methods assume vocabularies across sources are similar or even identical; which unfortunately stands in sharp contrast to the fact that many sources barely overlap. Further, these methods encode a meta-embedding for each word by reconstructing its actual embedding values (*word-encoder*), while valuable information of relationships (distances) among words within each source are not directly considered. In this work, we instead propose a novel *relation-encoder* learning approach called Similarity-Preserving Meta-Embedding (SimME) that directly integrates word-pair relationships from partially overlapping embedding sources. SimME embeds words such that their similarities are learned from those observed in multiple pre-trained sources. To handle relations between words that are *not* present in all sources, we introduce *maskout*, a new loss term, that steers the learning selectively to the sources containing said relations. SimME consistently outperforms state-of-the-art methods by 10% on average and with up to 20% across several core metrics in 4 popular mining tasks on 23 datasets.**

***Index Terms*—Meta-Embeddings, Text Mining, Word Representations, Semantics Preserving**

I. Introduction

Background and Motivation. Text mining has advanced rapidly in recent years as massive sites of unstructured text data is becoming common. For tackling space efficiency and modeling challenges, numerical representations of text, or *word embeddings* [1], [2], have become essential for text mining applications [3]–[5]. Traditional approaches, which use either one-hot encodings representing each word by a bit or frequency-based rules such as bag-of-words or tf-idf as representations to be fed into the machine learning models, all suffer from the curse of high dimensionality. Additionally, these representations do not meet the demand of applications from search engines, information retrieval, to sentiment analysis to capture word semantics and word relationships.

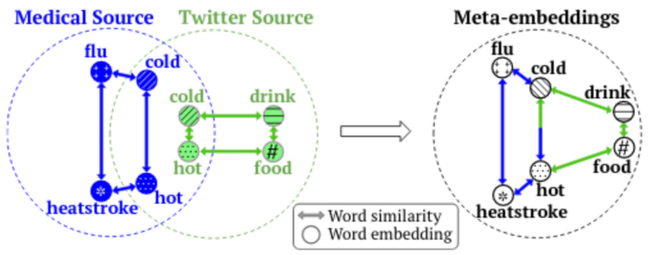


Fig. 1: Learning meta-embeddings from two partially overlapping embedding sources. Same words appear in all sources (*hot* and *cold*), yet their relative similarity may differ. Also, some may not present in all sources, leading to partially observed relations in only some source (*heatstroke* and *flu* or *food* and *drink*).

Hence, *word embedding* methods [2], [6] that create compact vector representations of words are popular. These embeddings not only reduce the dimensionality of text data but also aim to capture the inherent relationships among words via their relative distance in the embedding space. However, creating a set of embeddings requires a massive text corpus, substantial computational resources, and often prohibitively long training times. Given these excessive resource requirements, it has become common practice to download publicly-available *pre-trained* sets of embeddings [2], [6]–[8] provided by several organizations [9], [10]. These pre-trained sources have accelerated advances across a broad array of text mining tasks from sentiment analysis [4], keyphrase extraction [5], to digital healthcare [11].

Each available source captures unique semantics depending on the text upon which they were trained. The choice of source then strongly impacts task performance [3], [12]. Meanwhile picking the right source for a task is difficult as it is unclear which sources encode language usage in a task best. Consider, in Figure 1, a task using Twitter to support clinical diagnosis [13]. We have many options to pick embeddings from trained on tweets or medical text, but none trained on both. Which source should we choose? Assume our task involves the 6 words {*flu*, *heatstroke*, *cold*, *hot*, *drink*, *food*}, each source contains different subsets of this 6 word set. Neither source by itself can fully meet our need for these 6 words, while their combination would support us perfectly.

Fortunately, *meta-embedding* has recently been proposed as a mechanism to integrate multiple embedding sources into one embedding space [14]–[18] to aim to cover the needed vocabulary for a task. However, successful meta-embeddings should retain all word relationships inherent in these sources - complicating this meta-embedding learning process. Consider the additional example in Figure 1, where some word relationships are not consistent between sources such as *hot* and *cold*: being similar in Twitter while far apart in the Medical source. Moreover, some words appear in only one source, while being out-of-vocabulary (OOV) with respect to the other source such as *flu*, *heatstroke*, *drink*, and *food*.

State-of-the-Art. Recent methods learn meta-embeddings by reconstructing each word's raw embedding vector from pre-trained sources [14]–[18]. They focus on retaining encoded values of *words* individually, which we now call *word-encoder* methods. However, projecting such vectors into a new meta-embedding space does not necessarily preserve their *relative distance* within the original sources. Yet this relative distance is the key advantage in pre-trained sources since they were explicitly trained to capture the *relations between words* as a distance metric. Thus, meta-embeddings must carefully merge these informative relations into the united embedding space, meaning, a successful meta-embedding algorithm should be a *relation-encoder* rather than just a *word-encoder*.

Further, the *word-encoder* approaches are not effective for learning meta-embeddings for out-of-vocabulary (OOV) words. Rather, they simply adopt a single default embedding for *all* OOV words in each source. This limits their capacity since training is wasted on artificial embeddings that encode no data-driven information. Yet this OOV case is extremely important in practice because the vocabularies of publicly-available sources barely overlap, leading to a large number of these OOV cases as demonstrated in Section 4.

The closest work to ours [18] learns a meta-embedding for each word by incorporating information of its neighboring words or local neighborhoods. In contrast, we consider relationships of all words used in the target task. Their method indirectly preserves the relative distance of the neighbor words by assigning weights to each neighbor embeddings while ours directly preserves the pairwise distance among words.

Challenges. Our work targets two challenges in integrating semantics from partially-overlapping embedding sources.

• *Preservation of word similarity*. To produce a new meta-embedding space, directly projecting embeddings into such a joint space is not guaranteed to preserve the meaningful *relative distance between words*. It is difficult to accurately preserve these *word-pair similarities* in the new learned space. Moreover, similarities of a word-pair observed in multiple sources may contradict one another, mandating a learning approach to handle conflicts. No work so far directly preserves this crucial information.

• *Out-of-vocabulary (OOV) words*. The vocabulary covered by each pre-trained source is entirely dependent on its training corpus, leading to many words that do not exist in all sources (OOV words). For example, *flu*/*heatstroke* and *drink*/*food* in Figure 1, each appears in only one source. Meta-embeddings may become biased since they might be directed to skewed information from the sources that do not contain the words. Thus, this incomplete information complicates the effectiveness of meta-embeddings. Worse yet, the vocabularies of publicly-available sources barely overlap in practice, creating large numbers of OOV cases to be considered.

Proposed Method. To handle the aforementioned challenges, we propose the meta-embedding learning method, called **Sim**ilarity-Preserving **M**eta-**E**mbedding (SimME), that directly preserves word-pair relationships from multiple pre-trained embedding sources which have partially-overlapping vocabularies. SimME consists of two components: (1) A *Relation-Encoder* that learns an embedding space that maps a word-pair to their meta-embeddings. This neural network produces embeddings that are roughly equidistant in the original embedding space *and* the meta-embedding space. When one word is present in multiple sources, the *Relation Encoder* learns to appropriately balance between multiple distances. In contrast to *Word-Encoder* approaches [14], [16], [17], our *Relation-Encoder* ensures that the crucial information from the sources, the *relationships between words*, is directly maintained. (2) *Maskout*, an objective function for learning meta-embeddings, which employs an indicator function that allows the model to adaptively train meta-embeddings based only on the sources in which word-pair relationships are actually observed. *Maskout* successfully avoids uninformative artificial OOV embeddings [14], [17], instead allowing for fully data-driven meta-embeddings.

Contributions. Our contributions are as follows:

• We propose a novel training paradigm for meta-embeddings that successfully captures the relation between words through the relative distance among their embeddings across disparate embedding sources.

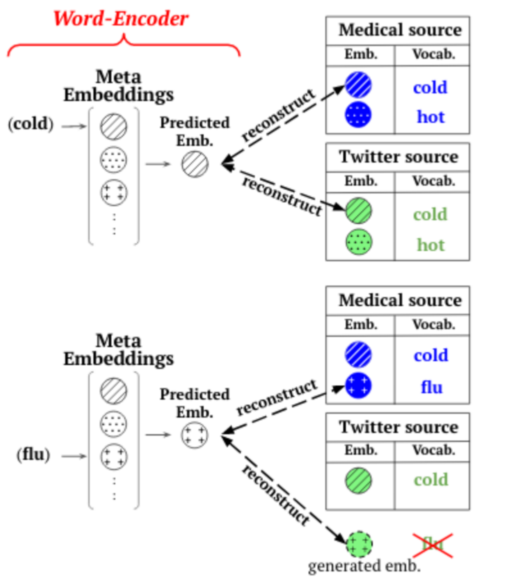
• We introduce the *maskout* optimization that allows our model to selectively learn from the sources containing the word-pair relations, elegantly handling prevalent OOV cases.

• SimME is evaluated on 23 datasets covering four different text-mining tasks. For each dataset, we evaluate the quality of learned meta-embeddings using four unique combinations from three popular pre-trained sources. SimME consistently outperforms 7 state-of-the-art baselines in all tasks, achieving 10% improvement on average and up to 20% in many cases.

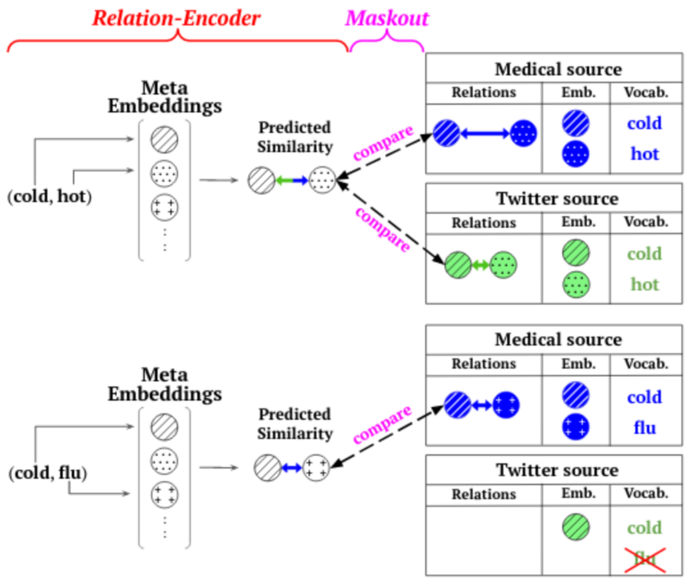
II. Related Work

Pre-trained word embeddings play a key role in standard practices across many text-mining tasks. Such embeddings are trained using one of many proposed methods [2], [6], [9], [19] and are often made publicly available [2], [7], [20], [21].

However, careful choice of the embedding sources has classically been crucial since each source contains vastly



(a) Word-based encoder. For OOV words, these methods generate the raw-values of unobserved embeddings [14], [16], [17].



(b) Relation-based encoder (ours). For word-pairs related to OOV, our method is able to focus only on real relationships without generating non-existing embeddings for OOV words.

Fig. 2: Comparison of the learning process for meta-embeddings.

different characteristics and shows variable performance across tasks [12]. In many cases, it remains unclear which embedding source is the best choice for a particular task, regardless of the domain. For example, despite training on a medical corpus, [22] demonstrate that domain-specific embeddings can be outperformed by general-purpose pre-trained embeddings on a clinical task. Similarly, [3] show the quality on diagnosis tasks varying dramatically across different embedding sources.

Several works with their focus on an individual task show that using multiple pre-trained embedding sources together improves their task's performance over using each separately. [23] demonstrate on named entity recognition that combining Brown clustering, CW [19] and HLBL [24] embeddings drives higher accuracy traded by a significant increase in the number of training parameters. Similarly, [25] shows an improvement of a part-of-speech tagging system while [26] focus on dependency-parsing. These works show an incremental benefit gained by combining multiple publicly available embedding sources. However, each work studies only a specific task that cannot generalize to other tasks. Moreover, they only simply ensemble those embedding sources rather than introducing a sophisticated method in integrating such embedding sources.

In contrast to depending on any specific tasks, *meta-embedding* learns a new integrated space of word embeddings from existing embedding sources with the aim of using them across many tasks. Recent works on meta-embedding [14]–[18] narrowly focus on *word-encoder* methods that reconstruct raw embedding vectors from multiple sources. The simplest approaches are through averaging [16] and concatenation [14]. More powerful are *1toN* and *1toN+* [14], which use a shallow neural network trained to predict meta-embeddings by minimizing the difference between the values of their meta-embeddings and the corresponding raw embeddings in original sources that are trained by different methods and different corpus. The linear methods proposed by [15] are similar to *1toN* except that they focus on creating a meta-embedding space from the embedding sources that are trained by the same algorithms on the same corpus, *i.e.*, they only differ on the weight initialization.

[18] generate meta-embeddings for each word based on linear weighted sums of its k-nearest neighboring words, requiring hand-selection of k-nearest parameters. Recently, *AEME* and its variants [17] use an autoencoder to reconstruct raw embeddings. For OOV words, both *1toN* and *AEME* first create random embeddings updated throughout training. These random embeddings do not bring any useful information into the meta-embeddings. None of these methods directly considers the similarities between word embeddings inherent to the original embedding sources which were carefully encoded during training.

III. Methodology

A. Problem Definition

Given a target vocabulary set $\mathcal{V}$ of t unique words denoted as $(w_1, \ldots, w_t)$, we aim to train a *meta-embedding space* $\mathbf{M}$ that contains embeddings corresponding to each word in $\mathcal{V}$.

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ be a set of word-pairs such that the unique words in $\mathcal{P}$ exactly match those in $\mathcal{V}$ where p_i denotes a pair of words. Let d be a pre-defined dimension of meta-embeddings in $\mathbf{M}$. For each $w_k \in \mathcal{V}$, $\mathbf{e}_k$ denotes the d-dimensional meta-embedding of w_k. Thus, $\mathbf{M} = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_t)$ and its dimension is $t \times d$. Given $\mathcal{S} = \{\mathbf{S}^1, \mathbf{S}^2, \ldots, \mathbf{S}^m\}$, a set of pre-trained embedding sources where their dimensions may differ, $\mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^m$ are the vocabularies covered by each source, respectively. For the j^{th} embedding source where $j \in \{1, 2, \ldots, m\}$, we measure

TABLE I: Basic Notation.

Notation	Description
$\mathcal{S}$	A set of pre-trained embedding sources; $\lvert\mathcal{S}\rvert = m$.
m	The numbers of pre-trained embedding sources.
$\mathbf{S}^j$	jth pre-trained embedding source in $\mathcal{S}$.
$\mathcal{V}^j$	Vocabulary covered by $\mathbf{S}^j$.
$\mathbf{M}$	Meta-embedding space.
$\mathcal{V}$	Vocabulary to train meta-embeddings in $\mathbf{M}$; $\lvert\mathcal{V}\rvert = t$.
t	The numbers of words in $\mathcal{V}$.
d	Dimension of word embeddings in $\mathbf{M}$.
w_k	A word k in $\mathcal{V}$.
$\mathbf{x}_k$	One-hot encoded of w_k in $\mathcal{V}$.
$\mathbf{e}_k$	Embedding of w_k in $\mathbf{M}$.
$\mathcal{P}$	A set of word-pairs consisting of words in $\mathcal{V}$; $\lvert\mathcal{P}\rvert = n$.
n	The numbers of word-pairs in $\mathcal{P}$.
p_i	ith word-pair in $\mathcal{P}$.
y_{p_i}, $y^j_{p_i}$	Similarity score of p_i measured in $\mathbf{M}$ and $\mathbf{S}^j$ respectively.

similarity score of two words in p_i by computing cosine similarity between their corresponding embeddings in that source, denoted by $y^j_{p_i}$, if both words in p_i appear in $\mathbf{S}^j$.

Our target is to generate t meta-embeddings in $\mathbf{M}$ that preserve the word-pair relationships observed in $\mathcal{S}$. That is, we learn meta-embeddings such that y_{p_i}, the similarity score of word-pair p_i in the meta-embedding space $\mathbf{M}$ measured by cosine similarity, is as close to $y^1_{p_i}, y^2_{p_i}, \ldots, y^m_{p_i}$ as possible. In the case that p_i is not present in all sources, similarity from only the sources that include both words should be preserved. For example, if word-pair p_i appears only in $\mathbf{S}^1$ and $\mathbf{S}^3$, y_{p_i} should be similar to only $y^1_{p_i}$ and $y^3_{p_i}$.

B. Similarity-Preserving Meta-Embedding

We propose **Sim**ilarity-Preserving **M**eta-**E**mbedding or SimME, a novel method for combining pre-trained embedding sources into one meta-embedding space $\mathbf{M}$. SimME generalizes across text-related tasks that need word representations for a set of vocabulary to be used in the task. $\mathbf{M}$ produces meta-embeddings for such vocabulary set that directly preserve the observed word-pair similarities in a given set of pre-trained sources. SimME consists of two components: a *Relation-Encoder* network, which encodes pairwise word similarity, and *Maskout*, a loss function that focuses the training of the *Relation-Encoder* on only the sources containing observed similarity, handling pairwise relationships for OOV words.

1) Training Data: SimME trains meta-embeddings directly over word-pair relationships. Thus, for a given text-related task, we first extract a set of word-pairs, $\mathcal{P}$, that serve as the input to SimME. The time complexity of the model linearly depends on the number of these input word-pairs, *i.e.*, $\lvert\mathcal{P}\rvert$. In practice, the number of word-pairs could increase exponentially if we consider all possible pairwise combination from a given vocabulary list. This incautious specifying too many uninformative word-pairs may require significantly unnecessary compute time.

We suggest to extract word-pairs according to the useful word relations for a given task. For example on a text classification, where each instance composes of a document that needs to be classified, each instance needs relationships among words in its document. Thus, we should generate all pairwise combinations of unique words appearing in each instance. Then the final set $\mathcal{P}$ is the composition of word-pairs merged from all instances. It is worth noting that most word-pairs are duplicates in many instances and so maintaining only the unique word-pairs would be more efficient and tractable.

2) Relation-Encoder: First, the Relation-Encoder network maps one-hot encoded vectors for word-pairs into a latent space via a trainable space $\mathbf{M}$, as illustrated in Figure 2b. The size of $\mathbf{M}$ is $t \times d$, forming by the size of a pre-defined target vocabulary set using in a given task, $\lvert\mathcal{V}\rvert = t$, and a pre-defined dimension of the meta-embeddings, d.

Let p_i represent a pair of words (w_k, w_l). Each word in p_i is represented using t-dimensional one-hot encodings, denoted by $\mathbf{x}_k$ and $\mathbf{x}_l$. Both encodings are then fed into the Relation-Encoder as pairs and projected into d-dimensional latent vectors $\mathbf{e}_k$ and $\mathbf{e}_l$, respectively, via an affine transformation as shown in Equation 1 where $\mathbf{b}$ is a bias vector.

$$\mathbf{e}_k = \mathbf{M}\mathbf{x}_k + \mathbf{b} \tag{1}$$

The embedded vectors or meta-embeddings $\mathbf{e}_k$ and $\mathbf{e}_l$ are contained in the weight matrix $\mathbf{M}$ and are learned throughout training. Thus, the embeddings of w_k and w_l exist together in the same space.

Subsequently, the similarity score between $\mathbf{e}_k$ and $\mathbf{e}_l$ corresponding to word-pair p_i is measured in space $\mathbf{M}$ using the cosine similarity. This similarity score is denoted by y_{p_i}.

$$y_{p_i} = \frac{{\mathbf{e}_k}^\top \cdot \mathbf{e}_l}{\lVert\mathbf{e}_k\rVert \cdot \lVert\mathbf{e}_l\rVert} \tag{2}$$

where $\lVert\rVert$ indicates cardinality and $\cdot$ is the dot product.

SimME preserves word similarities in the original pre-trained sources into its meta-embeddings. To achieve this, in each source that contains both w_k *and* w_l in word-pair p_i, we compute the cosine similarity between their corresponding embeddings in such source, resulting in a collection of similarity scores for the pair p_i corresponding to the number of sources. This collection is denoted by $y^1_{p_i}, y^2_{p_i}, \ldots, y^m_{p_i}$ from m pre-trained sources respectively. All parameters in $\mathbf{M}$ are trained to minimize the mean squared error between the similarity in the meta-embedding space and such collection of similarity scores, as shown in Equation 3 where m is the number of pre-trained sources and n is the number of word-pairs.

$$J(\mathbf{M}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(y_{p_i} - y^j_{p_i}\right)^2 \tag{3}$$

3) Maskout: Many words are rarely observed in all pre-trained sources (OOV words), resulting in various word-pairs related to them. For example in Figure 1, *flu* and *heatstroke* exist in the Medical Source but not in the Twitter Source. Both words are OOV words which related to several word-pairs, *e.g.*, (*flu*, *heatstroke*), (*hot*, *heatstroke*) and (*cold*, *flu*). We refer to these pairs as *nonmutual* and propose a novel loss term for meta-embedding, *maskout*, to handle these cases during the training. *Maskout* directs the training of the Relation-Encoder to solely preserve the observed similarity scores from only the sources that contain such *nonmutual* relation. This is in contrast to recently-proposed alternatives [14], [17], which create synthetic vectors for these OOV words in the sources that the words do not exist and subsequently are used to train their meta-embeddings.

As illustrated in Figure 2b, in the presence of nonmutual relationships, the weights of SimME are flexibly updated solely based on the sources that actually contain the relations according to the following modification of Equation 3:

$$J(\mathbf{M}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbb{1}_{p_i \in \mathbf{S}^j} \left(y_{p_i} - y_{p_i}^{j} \right)^2 \tag{4}$$

where p_i is a word-pair in $\mathcal{P}$ and $\mathbb{1}$ is an indicator function:

$$\mathbb{1}_{p_i \in \mathbf{S}^j} = \begin{cases} 1 & \text{if } p_i \in \mathbf{S}^j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Thus, for nonmutual relations, SimME minimizes the difference between predicted and observed similarity scores *only* in the sources where word-pairs are truly observed. In the case of mutual relationships, or those where a relationship exists in all sources, SimME preserves all similarity scores from all sources with equal weight. In principle, SimME learns to preserve the consensus of word-pair relationships among their truly observed information that are encoded in multiple pre-trained embedding sources.

IV. Experiments

We compare the quality of our proposed SimME against alternative methods on four text mining tasks using 23 datasets. For each experiment, we investigate all combinations of 3 sources in IV-A yielding in 4 different meta-embedding spaces.

A. Pre-Trained Embedding Sources

We use three popular publicly available sources as different semantic origins to be integrated into meta-embeddings:

• ***Wikipedia*** [2] contains 400,000 word embeddings pre-trained on Wikipedia 2014 and Gigaword 5 using GloVe [2]. All embeddings have 200 dimensions.

• ***BioMed*** [7] contains 200-dimensional embeddings for 2.4 million words, trained from over five billion words in PubMed and PubMed Central using Word2Vec [6].

• ***Twitter*** [2] provides 1.2 million 200-dimensional embeddings, trained on 27 billion words from tweets using GloVe.

B. Meta-Embedding Spaces

We assess four possible meta-embedding spaces resulting from integrating different semantics from large resources in IV-A. All methods are trained to produce these following four combinations to evaluate the compared performance:

• **Wiki-Bio.** This meta-embedding space integrates medical semantics from the *BioMed* source with general semantics from the *Wikipedia* source. The two sources have 124,171 overlapping words, or 4.7% of 2.6 million words found in their vocabulary union.

• **Bio-Twitter.** This meta-embedding space integrates colloquial semantics from the *Twitter* source with the medical semantics from *BioMed* source. The two original sources overlap for 1.8% of 3.5 million distinct words.

• **Wiki-Twitter.** This meta-embedding space integrates general semantics from two different styles of language uses in *Wikipedia* and *Twitter*. The two sources increment vocabulary coverage to 1.4 million unique words, overlapping by 10.4%.

• **Wiki-Bio-Twitter.** The final meta-embedding space integrates three different semantics from general semantics in *Wikipedia* to colloquial semantics used in *Twitter* and also medical semantics from the *BioMed*. The vocabulary in these three sources overlaps for 1.6% while its coverage is driven to 3.7 million words.

C. Compared Methods

We compare SimME to the following methods:

• ***Individual Sources.*** Each embedding source is used in isolation, forming a baseline to observe the impact of combining multiple sources. No multi-source information is encoded in this method.

• ***Average* [16].** Embeddings of the same word from sources are averaged. This requires equal dimension across sources.

• ***Concat: Concatenation* [14].** Embeddings of the same word in all sources are concatenated. The embeddings' dimension increases proportionally to the number of sources used.

• ***LLE: Locally Linear Meta-Embedding* [18].** The meta-embedding of each word is constructed by linearly weighted sum of its k-nearest neighbor embeddings. This cannot fully capture explicit information of all word-pair relationships, that need to be used in a task, from sources into the united space.

• ***AAEME: Averaged Auto-encoded Meta-Embedding* [17].** This method uses an encoder to encode raw embeddings of each word separately from each source. All these encoded outputs are then averaged to create a meta-embedding of such word. This meta-embedding is then trained to decode or reconstruct back to its raw embeddings in original sources.

• ***CAEME: Concatenated Auto-encoded Meta-Embedding* [17].** Similar to AAEME, the encoded outputs are now concatenated. This concatenation is treated as a meta-embedding, trained to decode its raw embeddings across sources.

• ***DAEME: Decoupled Auto-encoded Meta-Embedding* [17].** Similar to CAEME, the encoded outputs are concatenated but each part of the concatenation is separately trained to decode its raw embedding in the corresponding source.

Original sources treat OOV words by assigning to them the special unknown-token embedding, which also influences to the average and concatenation methods, *i.e.*, if a word does not appear in any source, the unknown-token embedding would be used to represent such word and fed to either averaging or concatenating operations in each of the two methods. LLE naturally overcomes the issue of OOV words since every word is reformed separately from its own neighboring words. The other state-of-the-art methods randomly generate embeddings for OOV words in the original sources whenever those words are not present. These artificial embeddings of OOV words are used to train the meta-embeddings while they themselves are simultaneous trained along with the meta-embeddings. This creates ingenuine information in OOV embeddings that would be integrated into their meta-embeddings.

TABLE II: Semantics similarity prediction results: correlation (rank) – higher indicates better embedding space in capturing word similarity as close as human judgement. Average rank shows performance overview across all datasets. **Bold**: best scores.

Datasets / Methods	Card	SimVerb	WS	RW	SimLex	WN-lap 2%	WN-lap 4%	WN-lap 6%	WN-lap 8%	WN-lap 10%	Ave. Rank
WIKI-BIO											
Wiki	0.10 (7)	0.20 (7)	0.51 (6)	0.33 (5)	0.34 (7)	0.49 (4)	0.49 (4)	0.49 (2)	0.48 (3)	0.48 (2)	4.7
Biomed	0.12 (2)	0.15 (8)	0.48 (8)	0.24 (7)	0.29 (8)	0.18 (8)	0.19 (8)	0.21 (8)	0.22 (8)	0.22 (8)	7.3
Average	0.10 (5)	0.20 (6)	0.55 (4)	0.33 (6)	0.34 (6)	0.51 (3)	0.49 (3)	0.48 (4)	0.47 (4)	0.46 (4)	4.5
Concat	0.10 (4)	0.21 (5)	0.54 (5)	0.34 (4)	0.35 (4)	0.51 (2)	0.49 (2)	0.49 (3)	0.49 (2)	0.47 (3)	3.4
LLE	0.03 (9)	0.23 (2)	0.50 (7)	0.21 (9)	0.35 (5)	0.09 (9)	0.08 (9)	0.08 (9)	0.08 (9)	0.08 (9)	7.7
AAEME	0.09 (8)	0.11 (9)	0.33 (9)	0.22 (8)	0.20 (9)	0.28 (7)	0.28 (7)	0.29 (7)	0.29 (7)	0.29 (7)	7.8
CAEME	0.10 (3)	0.22 (3)	0.57 (2)	**0.37** (1)	**0.36** (1)	0.47 (5)	0.46 (6)	0.45 (6)	0.44 (6)	0.43 (6)	3.9
DAEME	0.10 (6)	0.21 (4)	0.57 (3)	0.36 (2)	0.35 (3)	0.47 (6)	0.46 (5)	0.46 (5)	0.45 (5)	0.45 (5)	4.4
SimME	**0.14** (1)	**0.24** (1)	**0.58** (1)	0.34 (3)	0.36 (2)	**0.57** (1)	**0.56** (1)	**0.56** (1)	**0.55** (1)	**0.55** (1)	**1.3**
BIO-TWITTER											
Biomed	0.12 (2)	0.15 (5)	0.48 (7)	0.24 (3)	**0.29** (1)	0.45 (5)	0.45 (5)	0.45 (5)	0.45 (4)	0.46 (4)	4.1
Twitter	0.05 (8)	0.06 (8)	0.48 (6)	0.15 (8)	0.13 (8)	0.31 (8)	0.32 (8)	0.32 (8)	0.31 (8)	0.31 (8)	7.8
Average	0.04 (9)	-0.03 (9)	0.44 (8)	0.09 (9)	0.04 (9)	0.37 (7)	0.38 (7)	0.37 (7)	0.37 (7)	0.36 (7)	7.9
Concat	0.07 (6)	0.09 (6)	0.54 (4)	0.20 (5)	0.18 (6)	0.40 (6)	0.40 (6)	0.40 (6)	0.39 (6)	0.39 (6)	5.7
LLE	0.07 (7)	**0.19** (1)	0.54 (5)	0.15 (6)	0.26 (4)	0.04 (9)	0.03 (9)	0.04 (9)	0.03 (9)	0.03 (9)	6.8
AAEME	0.07 (5)	0.08 (7)	0.34 (9)	0.15 (7)	0.15 (7)	0.51 (3)	0.53 (3)	0.52 (3)	0.51 (3)	0.50 (3)	5.0
CAEME	0.09 (3)	0.16 (3)	0.58 (2)	0.25 (2)	0.27 (2)	0.55 (2)	0.53 (2)	0.53 (2)	0.51 (2)	0.51 (2)	2.2
DAEME	0.09 (4)	0.15 (4)	0.57 (3)	0.23 (4)	0.27 (3)	0.49 (4)	0.48 (4)	0.46 (4)	0.45 (5)	0.45 (5)	4.0
SimME	**0.14** (1)	0.18 (2)	**0.61** (1)	**0.27** (1)	0.26 (5)	**0.56** (1)	**0.55** (1)	**0.54** (1)	**0.53** (1)	**0.53** (1)	**1.5**
WIKI-TWITTER											
Wiki	0.10 (2)	0.20 (2)	0.51 (6)	**0.33** (1)	0.34 (2)	0.56 (2)	0.56 (2)	0.56 (2)	0.55 (2)	0.54 (2)	**2.3**
Twitter	0.05 (8)	0.06 (9)	0.48 (9)	0.15 (9)	0.13 (9)	0.04 (8)	0.06 (8)	0.08 (8)	0.08 (8)	0.11 (8)	8.4
Average	0.09 (5)	0.13 (8)	0.51 (7)	0.30 (4)	0.26 (5)	0.41 (7)	0.40 (7)	0.39 (7)	0.37 (7)	0.38 (7)	6.4
Concat	0.09 (4)	0.14 (7)	0.53 (5)	0.29 (5)	0.25 (7)	0.41 (6)	0.40 (6)	0.39 (6)	0.38 (6)	0.38 (6)	5.8
LLE	0.03 (9)	**0.25** (1)	0.54 (3)	0.20 (8)	**0.34** (1)	-0.02 (9)	-0.01 (9)	0.00 (9)	-0.01 (9)	-0.01 (9)	6.7
AAEME	0.08 (7)	0.19 (3)	**0.57** (1)	0.30 (3)	0.31 (3)	0.54 (3)	0.54 (3)	0.53 (3)	0.53 (3)	0.52 (3)	3.2
CAEME	0.09 (3)	0.16 (5)	0.54 (4)	0.28 (6)	0.28 (4)	0.52 (4)	0.51 (4)	0.50 (4)	0.48 (4)	0.47 (4)	4.2
DAEME	0.09 (6)	0.15 (6)	0.54 (2)	0.28 (7)	0.25 (6)	0.50 (5)	0.49 (5)	0.47 (5)	0.45 (5)	0.44 (5)	5.2
SimME	**0.15** (1)	0.17 (4)	0.50 (8)	0.31 (2)	0.25 (8)	**0.59** (1)	**0.60** (1)	**0.60** (1)	**0.58** (1)	**0.58** (1)	2.8
WIKI-BIO-TWITTER											
Wiki	0.10 (5)	0.20 (3)	0.51 (8)	0.33 (4)	0.34 (2)	0.56 (2)	0.55 (2)	0.55 (2)	0.55 (2)	0.54 (2)	3.2
Biomed	0.12 (2)	0.15 (7)	0.48 (10)	0.24 (8)	0.29 (7)	0.34 (8)	0.34 (8)	0.35 (8)	0.34 (8)	0.35 (8)	7.4
Twitter	0.05 (9)	0.06 (10)	0.48 (9)	0.15 (10)	0.13 (10)	0.06 (9)	0.06 (9)	0.07 (9)	0.09 (9)	0.09 (9)	9.3
Average	0.09 (7)	0.14 (9)	0.55 (7)	0.30 (7)	0.26 (9)	0.42 (7)	0.40 (7)	0.39 (7)	0.38 (7)	0.37 (7)	7.4
Concat	0.09 (6)	0.15 (8)	0.55 (5)	0.30 (6)	0.26 (8)	0.42 (6)	0.40 (6)	0.40 (6)	0.39 (6)	0.37 (6)	6.3
LLE	0.04 (10)	**0.24** (1)	0.55 (6)	0.19 (9)	0.34 (3)	-0.01 (10)	0.00 (10)	-0.01 (10)	-0.01 (10)	-0.01 (10)	7.9
AAEME	0.11 (4)	0.22 (2)	**0.66** (1)	**0.37** (1)	**0.35** (1)	0.51 (3)	0.50 (3)	0.50 (3)	0.50 (3)	0.49 (3)	2.4
CAEME	0.11 (3)	0.19 (4)	0.62 (3)	0.35 (2)	0.33 (4)	0.46 (4)	0.45 (4)	0.44 (4)	0.44 (4)	0.43 (4)	3.6
DAEME	0.09 (7)	0.15 (5)	0.62 (2)	0.31 (4)	0.30 (5)	0.44 (4)	0.42 (4)	0.42 (4)	0.42 (4)	0.41 (4)	4.3
SimME	**0.18** (1)	0.19 (5)	0.62 (4)	0.35 (3)	0.30 (5)	**0.61** (1)	**0.61** (1)	**0.61** (1)	**0.60** (1)	**0.60** (1)	**2.3**

D. Implementation Details

In each task, SimME is trained to produce 200-dimensional meta-embeddings for the vocabulary to be used in such task. The model is randomly initialized and trained for 200 epochs with a learning rate of 15 using stochastic gradient descent. Code for our proposed method is implemented in PyTorch and made publicly available[1].

E. Evaluation Tasks and Results

SimME is evaluated on a large series of datasets for four text mining tasks: semantics similarity prediction, synonym detection, text classification, and concept categorization.

1) Semantics Similarity Prediction: A common approach to evaluating the quality of word embeddings is to measure the degree to which they capture the semantic similarities of word-pairs as defined by human-rated scores [27]. First, the similarity of each word-pair in a dataset is measured by cosine similarity between their corresponding embeddings. Then we follow the standard practice [14], [17], using Spearman's Correlation Coefficient as a metric to measure how correlated the similarity between embeddings is to the given ground-truth similarities. Higher correlation indicates better performance.

We use 5 popular datasets with provided ground-truth similarities, including Card [28], SimVerb [29], WS [30], RareWords (RW) [31], and SimLex [32]. We further investigate the performance of meta-embeddings to more extreme cases when the datasets contain small overlap words among the original embedding sources by extracting further datasets from WordNet [33] where we can control the overlapping rate. We employ five additional datasets, each with a fixed overlapping rate of 2%, 4%, 6%, 8%, and 10% which we refer to as WN-lap 2%, WN-lap 4%, etc. Each word-pair in these datasets is assigned the ground-truth similarity by the Wu-Palmer similarity score defined in WordNet [33].

Table II shows the results of all methods on the four combinations integrated from the three pre-trained sources.

[1]`https://hiddenforreview.com`

TABLE III: Synonym detection results: AUC (rank). Higher AUC indicates synonymous words are better captured in an embedding space. Average rank shows performance overview across all datasets. **Bold**: best scores.

Methods \ Datasets	WordRep	WN-lap 2%	WN-lap 4%	WN-lap 6%	WN-lap 8%	WN-lap 10%	Ave. Rank
WIKI-BIO							
Wiki	0.66 (8)	0.79 (5)	0.78 (5)	0.77 (3)	0.76 (2)	0.76 (2)	4.2
Biomed	0.70 (4)	0.67 (8)	0.67 (8)	0.66 (8)	0.65 (8)	0.65 (8)	7.3
Average	0.68 (5)	0.80 (4)	0.78 (4)	0.77 (5)	0.75 (5)	0.74 (4)	4.5
Concat	0.68 (6)	0.80 (3)	0.79 (3)	0.77 (4)	0.75 (4)	0.74 (3)	3.8
LLE	0.65 (9)	0.54 (9)	0.54 (9)	0.54 (9)	0.54 (9)	0.54 (9)	9.0
AAEME	**0.72** (1)	0.74 (7)	0.73 (7)	0.71 (7)	0.70 (7)	0.68 (7)	6.0
CAEME	0.70 (3)	0.83 (2)	0.80 (2)	0.78 (2)	0.75 (3)	0.73 (5)	2.8
DAEME	0.67 (7)	0.75 (6)	0.73 (6)	0.72 (6)	0.70 (6)	0.69 (6)	6.2
SimME	0.71 (2)	**0.88** (1)	**0.87** (1)	**0.87** (1)	**0.86** (1)	**0.86** (1)	**1.2**
BIO-TWITTER							
Biomed	0.70 (4)	0.83 (4)	0.82 (2)	0.81 (2)	0.80 (2)	0.79 (2)	2.7
Twitter	0.62 (9)	0.61 (8)	0.61 (8)	0.60 (8)	0.60 (8)	0.59 (8)	8.2
Average	0.64 (7)	0.68 (7)	0.66 (7)	0.65 (7)	0.64 (7)	0.63 (7)	7.0
Concat	0.65 (6)	0.69 (6)	0.68 (6)	0.67 (6)	0.66 (6)	0.65 (6)	6.0
LLE	0.63 (8)	0.51 (9)	0.51 (9)	0.52 (9)	0.52 (9)	0.52 (9)	8.8
AAEME	**0.72** (1)	0.83 (3)	0.81 (3)	0.79 (3)	0.76 (4)	0.74 (4)	3.0
CAEME	0.70 (3)	0.83 (2)	0.80 (4)	0.78 (4)	0.76 (3)	0.75 (3)	3.2
DAEME	0.69 (5)	0.77 (5)	0.73 (5)	0.70 (5)	0.68 (5)	0.66 (5)	5.0
SimME	0.70 (2)	**0.85** (1)	**0.85** (1)	**0.84** (1)	**0.84** (1)	**0.84** (1)	**1.2**
WIKI-TWITTER							
Wiki	0.66 (3)	0.86 (2)	0.85 (2)	0.84 (2)	0.83 (2)	0.82 (2)	2.2
Twitter	0.62 (9)	0.49 (9)	0.48 (9)	0.48 (9)	0.46 (9)	0.47 (9)	9.0
Average	0.65 (5)	0.75 (7)	0.72 (7)	0.70 (7)	0.67 (7)	0.65 (7)	6.7
Concat	0.65 (6)	0.75 (6)	0.73 (6)	0.70 (6)	0.67 (6)	0.66 (6)	6.0
LLE	0.63 (7)	0.50 (8)	0.50 (8)	0.50 (8)	0.51 (8)	0.51 (8)	7.8
AAEME	**0.68** (1)	0.84 (3)	0.82 (3)	0.81 (3)	0.79 (3)	0.77 (3)	2.7
CAEME	0.66 (4)	0.82 (4)	0.79 (4)	0.77 (4)	0.73 (4)	0.71 (4)	4.0
DAEME	0.63 (8)	0.81 (5)	0.78 (5)	0.75 (5)	0.71 (5)	0.69 (5)	5.5
SimME	0.66 (2)	**0.91** (1)	**0.90** (1)	**0.89** (1)	**0.88** (1)	**0.87** (1)	**1.2**
WIKI-BIO-TWITTER							
Wiki	0.66 (8)	0.86 (2)	0.85 (2)	0.84 (2)	0.83 (2)	0.83 (2)	3.0
Biomed	0.70 (4)	0.74 (8)	0.73 (6)	0.73 (5)	0.73 (4)	0.72 (4)	5.2
Twitter	0.62 (10)	0.49 (10)	0.48 (10)	0.48 (10)	0.47 (10)	0.46 (10)	10.0
Average	0.67 (6)	0.76 (7)	0.73 (8)	0.71 (8)	0.69 (8)	0.66 (8)	7.5
Concat	0.67 (7)	0.76 (6)	0.73 (7)	0.71 (7)	0.69 (7)	0.67 (7)	6.8
LLE	0.64 (9)	0.50 (9)	0.50 (9)	0.50 (9)	0.51 (9)	0.51 (9)	9.0
AAEME	**0.72** (1)	0.82 (3)	0.80 (3)	0.79 (3)	0.77 (3)	0.75 (3)	2.7
CAEME	0.71 (3)	0.78 (4)	0.76 (4)	0.74 (4)	0.72 (5)	0.70 (5)	4.2
DAEME	0.69 (4)	0.77 (4)	0.74 (4)	0.73 (5)	0.71 (5)	0.69 (5)	4.5
SimME	0.71 (2)	**0.92** (1)	**0.91** (1)	**0.91** (1)	**0.90** (1)	**0.90** (1)	**1.2**

Across all datasets, SimME consistently achieves the top average ranking in three out of four meta-embeddings variations. The high correlation between SimME meta-embeddings similarities and human ratings across the vast majority of these datasets concludes that SimME successfully learns meta-embeddings in which the word-pair relationships are analogous to human-labeled ground truth similarities. On the WordNet datasets with small overlapping rate cases, SimME consistently outperforms the other alternatives. It achieves a correlation up to over 21% greater than the baselines, while alternative methods barely improve over the individual-source baselines. This concludes that SimME clearly improves word embeddings, especially for the case that a task contains a vocabulary with a small overlapping rate between sources.

2) Synonym Detection: Synonyms are commonly used to study relationships between words [27], [34]. Intuitively, semantic features shared by synonymous words should be reflected in the embedding space. We employ this task to evaluate to what degree this is true across all methods. Cosine similarity is used to measure word-pair relationships and is subsequently compared to the associated binary labels where the synonym word-pairs are considered as positive examples. The Area Under the ROC Curve (AUC) is calculated across all word-pairs in the evaluation dataset. Intuitively, a high AUC indicates that the similarity between a pair of words is high when the words are synonyms and is low otherwise.

We use a dataset from WordRep [35] by labelling the antonym word-pairs as the negative examples and labelling the similar word-pairs as the positive examples. We randomly downsample the positive examples to have the same size as the negative examples. Similar to the semantic similarity prediction task, we investigate further on the extreme cases when the datasets contains small overlapping words among the original embedding sources. Again, we use the small-overlap WordNet datasets. The synonym word-pairs are informed by *synsets* in WordNet and labeled as positive, while the nonsynonym pairs

TABLE IV: Results of text classification (left): %Accuary (rank), and concept categorization (right): %NMI (rank). Higher scores shows better performance of using an embedding space in each task. Average rank shows performance overview across all datasets. **Bold**: best scores.

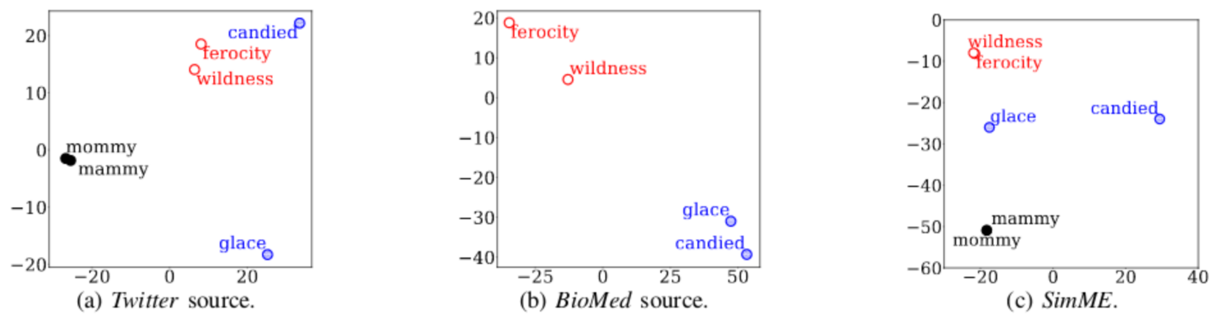
	Text Classification					*Concept Categorization*			
Methods \ Datasets	Airline	SubjObj	MR400	ProCon400	Ave. Rank	Bless	ESSLLI	SimVerb	Ave. Rank
Wiki-Bio									
Wiki	82.88 (4)	89.96 (3)	**59.20** (1)	86.70 (5)	3.3	0.37 (4)	16.67 (8)	1.99 (3)	5.0
Biomed	81.18 (6)	85.81 (9)	55.80 (5)	85.80 (6)	6.5	0.12 (8)	19.27 (5)	0.88 (9)	7.3
Average	83.40 (2)	90.45 (2)	55.80 (5)	87.10 (4)	3.3	0.42 (3)	19.05 (6)	1.44 (5)	4.7
Concat	83.30 (3)	**90.80** (1)	57.10 (3)	88.30 (2)	2.3	0.37 (4)	15.86 (9)	1.19 (8)	7.0
LLE	80.76 (7)	87.91 (6)	54.20 (7)	84.60 (8)	7.0	**0.46** (1)	**22.17** (1)	**2.55** (1)	**1.0**
AAEME	80.76 (7)	88.06 (5)	49.60 (9)	87.50 (3)	6.0	0.08 (9)	20.05 (2)	2.41 (2)	4.3
CAEME	79.60 (9)	87.66 (7)	56.30 (4)	74.20 (9)	7.3	0.17 (6)	19.28 (4)	1.54 (4)	4.7
DAEME	81.71 (5)	87.61 (8)	50.40 (8)	85.00 (7)	7.0	0.12 (7)	18.01 (7)	1.30 (6)	6.7
SimME	**84.78** (1)	88.26 (4)	57.50 (2)	**89.10** (1)	**2.0**	0.42 (2)	19.84 (3)	1.22 (7)	4.0
Bio-Twitter									
Biomed	81.18 (5)	85.81 (9)	55.80 (4)	85.80 (3)	5.3	0.12 (8)	19.27 (5)	0.88 (7)	6.7
Twitter	82.56 (3)	88.86 (2)	56.30 (3)	87.10 (2)	2.5	0.29 (5)	19.32 (4)	0.73 (9)	6.0
Average	80.13 (7)	**89.31** (1)	55.40 (5)	84.20 (4)	4.3	0.49 (2)	19.24 (6)	1.24 (5)	4.3
Concat	82.88 (2)	88.86 (2)	56.70 (2)	74.60 (7)	3.3	0.41 (4)	18.31 (8)	0.86 (8)	6.7
LLE	79.39 (9)	87.76 (7)	50.40 (7)	82.50 (5)	7.0	**0.53** (1)	**20.80** (1)	1.37 (4)	**2.0**
AAEME	79.92 (8)	88.11 (5)	50.00 (8)	72.10 (8)	7.3	0.17 (6)	20.62 (2)	**1.94** (1)	3.0
CAEME	82.14 (4)	87.96 (6)	49.60 (9)	60.40 (9)	7.0	0.11 (9)	18.01 (9)	1.75 (2)	6.7
DAEME	80.34 (6)	87.41 (8)	52.10 (6)	77.10 (6)	6.5	0.13 (7)	18.36 (7)	1.40 (3)	5.7
SimME	**83.72** (1)	88.41 (4)	**59.20** (1)	**88.70** (1)	**1.8**	0.44 (3)	20.62 (2)	1.13 (6)	3.7
Wiki-Twitter									
Wiki	82.88 (4)	89.96 (2)	59.20 (4)	86.70 (6)	4.0	0.37 (4)	16.67 (8)	**1.99** (1)	4.3
Twitter	82.56 (5)	88.86 (4)	56.30 (5)	87.10 (5)	4.8	0.29 (6)	19.32 (4)	0.73 (9)	6.3
Average	82.98 (3)	88.51 (5)	**60.00** (1)	85.00 (7)	4.0	0.35 (5)	16.23 (9)	0.75 (8)	7.3
Concat	83.72 (2)	**90.00** (1)	**60.00** (1)	85.00 (7)	2.8	0.42 (2)	17.01 (7)	1.00 (6)	5.0
LLE	79.60 (8)	87.11 (9)	42.50 (9)	82.50 (9)	8.8	**0.42** (1)	18.88 (5)	1.25 (3)	3.0
AAEME	78.96 (9)	87.91 (7)	50.00 (7)	**91.25** (1)	6.0	0.13 (9)	19.34 (3)	1.74 (2)	4.7
CAEME	81.50 (6)	87.66 (8)	43.75 (8)	88.75 (4)	6.5	0.15 (8)	**20.47** (1)	1.05 (5)	4.7
DAEME	80.13 (7)	88.16 (6)	56.25 (6)	**91.25** (1)	5.0	0.22 (7)	17.41 (6)	0.79 (7)	6.7
SimME	**83.83** (1)	88.96 (3)	**60.00** (1)	**91.25** (1)	**1.5**	0.38 (3)	**20.47** (1)	1.11 (4)	**2.7**
Wiki-Bio-Twitter									
Wiki	82.88 (2)	89.96 (3)	59.20 (3)	86.70 (7)	3.8	0.37 (5)	16.67 (10)	**1.99** (1)	5.3
Biomed	81.18 (7)	85.81 (10)	55.80 (6)	85.80 (8)	7.8	0.12 (10)	19.27 (2)	0.88 (9)	7.0
Twitter	82.56 (3)	88.86 (4)	56.30 (5)	87.10 (6)	4.5	0.29 (6)	**19.32** (1)	0.73 (10)	5.7
Average	82.35 (5)	**91.10** (1)	58.75 (4)	87.50 (5)	3.8	0.40 (4)	17.75 (9)	0.89 (8)	7.0
Concat	82.45 (4)	90.15 (2)	61.25 (2)	82.50 (9)	4.3	0.40 (2)	18.80 (4)	1.00 (6)	4.0
LLE	79.60 (9)	88.16 (6)	53.75 (7)	81.25 (10)	8.0	0.40 (3)	19.16 (3)	1.11 (5)	3.7
AAEME	80.76 (8)	86.21 (9)	51.25 (8)	91.25 (4)	7.3	0.23 (7)	18.01 (5)	0.97 (7)	6.3
CAEME	81.50 (6)	88.01 (7)	50.00 (9)	92.50 (3)	6.3	0.20 (9)	18.01 (5)	1.32 (3)	5.7
DAEME	79.28 (9)	86.86 (7)	43.75 (9)	**93.75** (1)	6.5	0.21 (7)	18.01 (5)	1.72 (2)	4.7
SimME	**85.41** (1)	88.66 (5)	**63.75** (1)	**93.75** (1)	**2.0**	**0.78** (1)	18.01 (5)	1.32 (4)	**3.3**

with negative labels are sampled randomly with the same size as the positive cases.

Our results in Table III report strong performance that SimME accomplishes the top average rank across all datasets in all combinations. This indicates that its learned embedding space is superior to alternatives in terms of projecting synonyms into the same neighborhoods. SimME consistently improves the mean AUC in all settings over all alternative methods by up to 20% and again clearly shows the success on the extreme cases on datasets with small overlapping rates. Moreover, unlike other methods, SimME shows that combining the complete three embedding sources, in comparison to combining any two sources, results in the strongest performance in infusing pairwise semantics into meta-embeddings by mapping the embeddings of synonym words into the meta-embedding space.

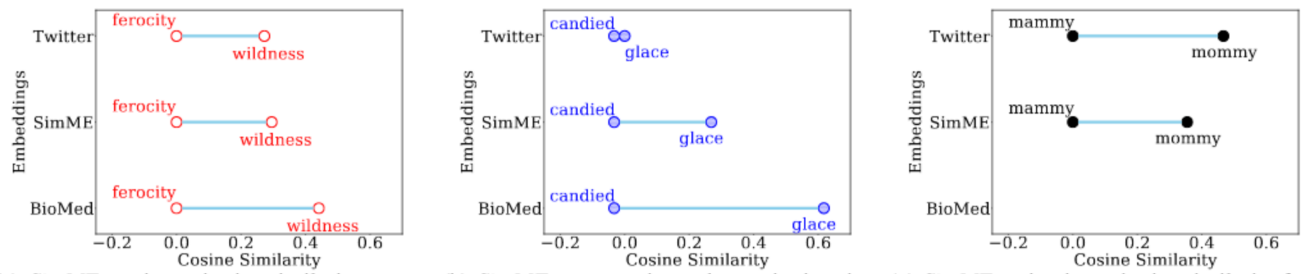
3) Text Classification: In this task, our goal is to evaluate the quality of SimME compared to other alternative methods on one of the most common text mining task. Each dataset contains balanced examples of binary labels. Except the different embedding methods used for comparison, the rest of the training pipeline remains the same in all experiments. We split the data into 0.7:0.1:0.2 for training:validating:testing. We use Long Short-Term Memory (LSTM) networks with 2 layers of 256 hidden units to train the model and report accuracy on testing data as the evaluation metric.

Four datasets are used in this task. The first dataset, referred to as Airline, is from public Kaggle Dataset: Twitter US Airline Sentiment, originally released by CrowdFlower. We use all 2,363 positive examples in the dataset and randomly sample negative examples to obtain the balanced data. Another dataset is the subjective-objective dataset [36], referred to as



(a) *Twitter* source. (b) *BioMed* source. (c) *SimME*.

Fig. 3: t-SNE projection of pre-trained word embeddings and their associated *SimME* meta-embeddings.



(a) *SimME* retains pairwise similarity when original sources agree on its similarity.

(b) *SimME* compromises when pairwise similarities disagree.

(c) *SimME* maintains pairwise similarity from only the observed embedding source.

Fig. 4: Cosine similarity scores between pairs of words.

SubjObj, containing 10,000 examples with 5,000 subjective labels drawn from the Rotten Tomatoes pages and 5,000 objective labels drawn from IMDb plot summaries. Another two datasets, referred to as MR400 and ProCon400, are used to observe the impact of different embedding methods on smaller datasets which usually worsen performances on downstream tasks. MR400 and ProCon400 are randomly downsampled, with balanced positive and negative labels, from movie reviews [37] and Pros-Cons reviews [38] respectively.

The left part in Table IV shows results of this task. SimME again reaches the top average rank across all datasets indicating that the meta-embeddings by SimME are completely beneficial on the most common task in text mining with the improved accuracy up to 6%. The performance on MR400 shows interesting results that SimME boosts the accuracy especially when combining the complete three sources while most methods suffer from training on small data. Similarly, SimME improves the performance in all settings on ProCon400.

4) Concept Categorization: Concept Categorization is another popular task for evaluating the quality of word embeddings [27]. Given a number of concepts, this task clusters word embeddings or word relations, depending on the dataset, into categories. For clustering tasks on word relations, we compute word-pair relations by commonly use of vector arithmetic between their embeddings [39], [40], *i.e.*, using the arithmetic subtraction of their corresponding embeddings for each pair. We then do clustering on these vectors. Our metric is Normalized Mutual Information (NMI).

Three standard datasets are used. Bless [41] contains 26,553 word-pairs labelled with 6 different relations including attribute, coordinate, event, hypernym, member, and random. ESSLLI [42] comes with 44 words grouped into 6 categories of animates and inanimates. Lastly, SimVerb [29] provides 3,500 word-pairs labelled by 5 classes of antonyms, cohyponyms, hypernyms, synonym, and none.

Results on this concept categorization are shown on the right part in Table IV. Across three datasets, SimME reaches the top average rank on two out of four combinations. When combining two sources, LLE performs very competitive to SimME. This shows that the local neighbor words separately in each source have a higher impact on creating meta-embeddings that project same concepts closed together. However, using more sources, *e.g.*, **WIKI-BIO-TWITTER**, their consensus on global relationship among words is effectively fused using SimME method that successfully improves the task.

F. Case Study Via Visual Examination

To better understand the relationships between words in the meta-embedding space trained by SimME, in Figure 3 we show three word-pairs in their original sources and in the meta-embedding space learned by SimME. Using t-SNE [43], we observe that "ferocity" and "wildness", which are close in both sources, remain, as expected, close in the meta-embedding space. Meanwhile "mommy" and "mammy", which appear only in the Twitter source, remain extremely close to one another in the meta-embedding space. Finally, "glace" and "candied" end up in between the observed relations.

Using cosine similarity, we show in Figure 4a that SimME successfully retains observed similarities when the sources agree on the distance between word embeddings. Then, when the sources disagree on the distance, as shown in Figure 4b, SimME embeds these words into a space where their similarity falls in between the observed similarities. Finally, for nonmutual word-pairs, shown in Figure 4c, SimME retains the similarity observed in the original source in which the word-pair exists. In this case, the distance itself is rarely identical to that of the source due to the relationships between these and other words during the learning process.

V. Conclusion

In this work, we propose the SimME meta-embedding method that transforms words into meaningful compact vectors – eliminating the curse of high dimensionality for leveraging massive unstructured text data. SimME explicitly preserves word-pair similarity when combining pre-trained embedding sources. To handle the pervasive challenge of out-of-vocabulary (OOV) words, SimME adopts a novel training procedure for meta-embeddings, called *maskout*, that is shown to result in selective preserving information only from the truly observed sources. This approach is effective as demonstrated in our experiments with a vastly superior performance on a plethora of datasets. By preserving word-pair relationships from relevant sources, our SimME method improves the quality of word embeddings in capturing semantic information while providing an avenue for the combination of non-overlapping vocabulary across sources.

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013, pp. 3111–3119.

[2] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of EMNLP*, 2014, pp. 1532–1543.

[3] J. Thadajarassiri, C. Sen, T. Hartvigsen, X. Kong, and E. Rundensteiner, "Comparing general and locally-learned word embeddings for clinical text mining," in *Proceedings of IEEE-BHI*, 2019, pp. 1–4.

[4] W. Xue and T. Li, "Aspect based sentiment analysis with gated convolutional networks," in *Proceedings of ACL*, 2018, pp. 2514–2523.

[5] S.-E. Lee, K.-M. Kim, W.-J. Ryu, J. Park, and S. Lee, "From text classification to keyphrase extraction for short text," in *Proceedings of Big Data*, 2019, pp. 1137–1142.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[7] S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou, "Distributional semantics resources for biomedical text processing," *Proceedings of LBM*, pp. 39–44, 2013.

[8] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of ACL*, 2014, pp. 238–247.

[9] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of ACL*, 2012, pp. 873–882.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[11] C. Sen, T. Hartvigsen, X. Kong, and E. Rundensteiner, "Patient-level classification on clinical note sequences guided by attributed hierarchical attention," in *Proceedings of Big Data*, 2019, pp. 930–939.

[12] Y. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena, "The expressive power of word embeddings," *arXiv preprint arXiv:1301.3226*, 2013.

[13] M. S. Deiner, S. D. McLeod, J. Chodosh, C. E. Oldenburg, C. A. Fathy, T. M. Lietman, and T. C. Porco, "Clinical age-specific seasonal conjunctivitis patterns and their online detection in twitter, blog, forum, and comment social media posts," *Investigative ophthalmology & visual science*, vol. 59, no. 2, pp. 910–920, 2018.

[14] W. Yin and H. Schütze, "Learning word meta-embeddings," in *Proceedings of ACL*, 2016, pp. 1351–1360.

[15] A. Muromägi, K. Sirts, and S. Laur, "Linear ensembles of word embedding models," in *Proceedings of NoDaLiDa*, May 2017, pp. 96–104.

[16] J. Coates and D. Bollegala, "Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings," in *Proceedings of NAACL-HLT (Short Papers)*, 2018, pp. 194–198.

[17] D. Bollegala and C. Bao, "Learning word meta-embeddings by autoencoding," in *Proceedings of COLING*, 2018, pp. 1650–1661.

[18] D. Bollegala, K. Hayashi, and K.-I. Kawarabayashi, "Think globally, embed locally: locally linear meta-embedding of words," in *Proceedings of IJCAI*, 2018, pp. 3970–3976.

[19] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of ICML*, 2008, pp. 160–167.

[20] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu, "Biowordvec, improving biomedical word embeddings with subword information and mesh," *Scientific data*, vol. 6, no. 1, pp. 1–9, 2019.

[21] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of LREC*, 2018.

[22] Y. Wang, S. Liu, N. Afzal, M. Rastegar-Mojarad, L. Wang, F. Shen, P. Kingsbury, and H. Liu, "A comparison of word embeddings for the biomedical natural language processing," *Biomedical Informatics*, vol. 87, pp. 12–20, 2018.

[23] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of ACL*, 2010, pp. 384–394.

[24] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of ICML*, 2007, pp. 641–648.

[25] Y. Tsuboi, "Neural networks leverage corpus-wide information for part-of-speech tagging," in *Proceedings of EMNLP*, 2014, pp. 938–950.

[26] M. Bansal, K. Gimpel, and K. Livescu, "Tailoring continuous word representations for dependency parsing," in *Proceedings of ACL*, 2014, pp. 809–815.

[27] A. Bakarov, "A survey of word embeddings evaluation methods," *arXiv preprint arXiv:1801.09536*, 2018.

[28] M. T. Pilehvar, D. Kartsaklis, V. Prokhorov, and N. Collier, "Card-660: Cambridge rare word dataset-a reliable benchmark for infrequent word representation models," in *Proceedings of EMNLP*, 2018, pp. 1391–1401.

[29] D. Gerz, I. Vulić, F. Hill, R. Reichart, and A. Korhonen, "Simverb-3500: A large-scale evaluation set of verb similarity," in *Proceedings of EMNLP*, 2016, pp. 2173–2182.

[30] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of WWW*, 2001, pp. 406–414.

[31] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *Proceedings of CoNLL*, 2013, pp. 104–113.

[32] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.

[33] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[34] H. Fei, S. Tan, and P. Li, "Hierarchical multi-task word embedding learning for synonym prediction," in *Proceedings of KDD*, 2019, pp. 834–842.

[35] B. Gao, J. Bian, and T.-Y. Liu, "Wordrep: A benchmark for research on learning word representations," *arXiv preprint arXiv:1407.1640*, 2014.

[36] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity," in *Proceedings of ACL*, 2004, pp. 271–278.

[37] ——, "Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of ACL*, 2005, pp. 115–124.

[38] M. Ganapathibhotla and B. Liu, "Mining opinions in comparative sentences," in *Proceedings of COLING*, 2008, pp. 241–248.

[39] A. Gittens, D. Achlioptas, and M. W. Mahoney, "Skip-gram – zipf + uniform = vector additivity," in *Proceedings of ACL*, Jul. 2017, pp. 69–76.

[40] C. Allen and T. Hospedales, "Analogies explained: Towards understanding word embeddings," in *Proceedings of ICML*, 2019, pp. 223–231.

[41] M. Baroni and A. Lenci, "How we blessed distributional semantic evaluation," in *Proceedings of GEMS*, 2011, pp. 1–10.

[42] M. Baroni, S. Evert, and A. Lenci, "Esslli 2008 workshop on distributional lexical semantics. hamburg, germany: Association for logic," *Language and Information*, 2008.

[43] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.