IDIAP RESEARCH REPORT

"THE SUM OF ITS PARTS": JOINT LEARNING OF WORD AND PHRASE REPRESENTATIONS WITH AUTOENCODERS

Rémi Lebret [a] Ronan Collobert

Idiap-RR-21-2015

JUNE 2015

[a] Idiap

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

"The Sum of Its Parts": Joint Learning of Word and Phrase Representations with Autoencoders

Rémi Lebret REMI@LEBRET.CH
Idiap Research Institute, Martigny, Switzerland
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Ronan Collobert[1] RONAN@COLLOBERT.COM
Facebook AI Research, Menlo Park, CA, USA
Idiap Research Institute, Martigny, Switzerland

Abstract

Recently, there has been a lot of effort to represent words in continuous vector spaces. Those representations have been shown to capture both semantic and syntactic information about words. However, distributed representations of phrases remain a challenge. We introduce a novel model that jointly learns word vector representations and their summation. Word representations are learnt using the word co-occurrence statistical information. To embed sequences of words (i.e. phrases) with different sizes into a common semantic space, we propose to average word vector representations. In contrast with previous methods which reported *a posteriori* some compositionality aspects by simple summation, we simultaneously train words to sum, while keeping the maximum information from the original vectors. We evaluate the quality of the word representations on several classical word evaluation tasks, and we introduce a novel task to evaluate the quality of the phrase representations. While our distributed representations compete with other methods of learning word representations on word evaluations, we show that they give better performance on the phrase evaluation. Such representations of phrases could be interesting for many tasks in natural language processing.

1. Introduction

Human language "makes infinite use of finite means" (Humboldt, 1836). A large number of sentences can be generated from a finite set of words. Thus there has been a lot of effort to capture the meaning of words. Some approaches are based on *distributional* word representations (Lund & Burgess, 1996; Patel et al., 1998), others are based on *distributed* representations (Bengio, 2008; Collobert et al., 2011; Mnih & Kavukcuoglu, 2013; Mikolov et al., 2013b; Lebret & Collobert, 2014; Pennington et al., 2014; Levy & Goldberg, 2014) where the meaning of a word is encoded as a vector computed from co-occurrence statistics of a word and its neighboring words. Finally, distributed representations emerged as the solution to many natural language processing (NLP) tasks (Turney & Pantel, 2010; Collobert et al., 2011).

Given these representations of words in a vector space, techniques for combining them have been proposed to get representations of phrases or sentences. These compositional models involve vector addition or multiplication (Mitchell & Lapata, 2010). Such simple compositions have shown to perform competitively on the paraphrase detection and phrase similarity tasks (Blacoe & Lapata, 2012). More sophisticated approaches use techniques from logic, category theory, and quantum information (Clark et al., 2008). Others use the syntactic relations between words to treat certain words as functions and other as arguments such as adjective-noun composition (Baroni & Zamparelli, 2010) or noun-verb composition (Grefenstette et al., 2013). Recursive neural network model for semantic compositionality has also been proposed (Socher et al., 2012), where each word has a matrix-vector representation: the vector captures its meaning (as it is initialized with a pre-trained distributed representation), while the matrix learns throught a parse tree how it modifies the meaning of the other word that it combines

[1] *All research was conducted at Idiap Research Institute, before Ronan Collobert joined Facebook AI Research.*

with. Many recent works are based on distributed representations of phrases to tackle a wide range of application in NLP: machine translation (Bahdanau et al., 2015), constituency parsing (Legrand & Collobert, 2015), sentiment analysis (Socher et al., 2013), or image captioning (Lebret et al., 2015). There is therefore a clear need for distributed word representations that can be easily extrapolated to meaningful phrase representations.

We argue that distributed representation and composition must go hand in hand, i.e., they must be mutually learned. We present a model that learns to capture meaning of words in distributed representations using a low-rank approximation of a large word co-occurrence matrix. We choose to stochastically perform this low-rank approximation which enables the model to simultaneously train these representations to compose for producing representations of phrases (see Figure 1). As composition function, we choose a simple weighted addition for its simplicity and for enabling sequences of words with different lengths to be representated in a common vector space. Aside from generating distributed representations of words and phrases, this model gives an encoding function (represented by a matrix) which can be used to encode new words or even phrases based on their co-occurrence counts. This offers two different alternatives for phrase representations: (1) representation for a query phrase can be inferred by averaging vector representations of its words (only if they all were in the training set), or (2) by using its word co-occurrence statistics.

Evaluation on the popular word similarity and analogy tasks demonstrate the capability of our joint model for capturing good distributed representations. We then introduce a novel task for evaluating phrase representations. Given a phrase representation, the objective is to retrieve the words that compose the phrase. We compare our model against other state-of-the-art methods for distributed word representations which capture meaningful linear substructures (Mikolov et al., 2013a; Pennington et al., 2014). We show that our model achieves similar performance on word evaluation tasks, but that it outperforms other methods on the phrase evaluation task.

2. Related Works

In the literature, two major model families exist for learning distributed word representations: the count-based methods and the predictive-based methods.

The count-based methods consist of using the statistical information contained in large corpora of unlabeled text to build large matrices by simply counting words (word cocoocurrence statistics). The rows correspond to words or terms, and the columns correspond to a local context. The context can be documents, such as in latent semantic analysis (LSA) (Deerwester et al., 1990); or other words (Lund & Burgess, 1996). To generate low-dimensional word representations, a low-rank approximation of these large matrices is performed, mainly with a singular value decomposition (SVD). Many authors proposed to improve this model with different transformations for the matrix of counts, such as positive pointwise mutual information (PPMI) (Bullinaria & Levy, 2007; Levy & Goldberg, 2014), or a square root of the co-occurrence probabilities in the form of Hellinger PCA (Lebret & Collobert, 2014). Instead of using the co-occurrence probabilities, (Pennington et al., 2014) suggest that word vector representations should be learnt with ratios of co-occurrence probabilities. For this purpose, they introduce a log-bilinear regression model that combines both global matrix factorization and local context window methods.

The predictive-based model has first been introduced as a neural probabilistic language model (Bengio et al., 2003). A neural network architecture is trained to predict the next word given a window of preceding words, where words are representated by low-dimensional vector. Since, some variations of this architecture have been proposed. (Collobert et al., 2011) train a language model to discriminate a two-class classification task: if the word in the middle of the input window is related to its context or not. More recently, the need of full neural architectures has been questioned (Mnih & Kavukcuoglu, 2013; Mikolov et al., 2013a). Mikolov et al. (2013a) propose two predictive-based log-linear models for learning distributed representations of words: (i) the continous bag-of-words model (CBOW), where the objective is to correctly classify the current (middle) word given a symmetric window of context words around it; (ii) the skip-gram model, where instead of predicting the current word based on the context, it tries to maximize classification of a word based on another word in the same sentence. In Mikolov et al. (2013b), the authors also introduce a data-driven approach for learning phrases, where the phrases are treated as individual tokens during the training.

In this paper, we leverage both families: (i) we use the statistical information for learning distributed word representations by approximating the Hellinger PCA with an autoencoder network; (ii) we jointly learn to predict the words that compose a given phrase.

3. A Joint Model

Some prior works have designed models to learn word representations (Mnih & Kavukcuoglu, 2013; Mikolov et al., 2013b; Lebret & Collobert, 2014), while others have proposed models to compose these word representations (Mitchell & Lapata, 2010; Socher et al., 2012). We propose instead to jointly learn word representations and

their composition by simple summation.

3.1. Learning Word Representations w.r.t. the Hellinger Distance

As words occurring in similar contexts tend to have similar meanings (Harris, 1954), word co-occurrence statistics are generally used to embed similar words into a common vector space (Turney & Pantel, 2010). Common approaches calculate the frequencies, apply some transformations (tf-idf, PPMI), reduce the dimensionality, and calculate the similarities. More recently, Lebret & Collobert (2014) proposed a novel method based on a *Hellinger PCA* of the word co-occurrence matrix. They showed that word representations can be learnt even with a reasonable number of context words. Inspired by this work, we propose to stochastically perform this low-rank approximation. For this purpose, we use an autoencoder with only linear activations to find an optimal solution related to the Hellinger PCA (Bourlard & Kamp, 1988). Replacing the PCA by an autoencoder allows us to learn jointly a cost function which constrains the word information to be kept by summation.

3.1.1. Word Co-Occurrence Probabilities

"You shall know a word by the company it keeps" (Firth, 1957). Keeping this famous quote in mind, word co-occurrence probabilities are computed by counting the number of times each context word $c \in \mathcal{D}$ (where $\mathcal{D} \subseteq \mathcal{W}$) occurs around a word $w \in \mathcal{W}$:

$$p(c|w) = \frac{p(c,w)}{p(w)} = \frac{n(c,w)}{\sum_{c_j \in \mathcal{D}} n(c_j,w)}\,, \tag{1}$$

where $n(c,w)$ is the number of times a context word c occurs in the surrounding of the word w. A multinomial distribution of $|\mathcal{D}|$ classes (words) is thus obtained for each word w:

$$P_w = \{p(c_1|w), \ldots, p(c_{|\mathcal{D}|}|w)\}\,. \tag{2}$$

3.1.2. Hellinger Distance

Similarities between words can be derived by computing a distance between their corresponding word distributions. Several distances (or metrics) over discrete distributions exist, such as the Bhattacharyya distance, the Hellinger distance or Kullback-Leibler divergence. We chose here the Hellinger distance for its simplicity and symmetry property (as it is a true distance). Considering two discrete probability distributions $P = (p_1, \ldots, p_k)$ and $Q = (q_1, \ldots, q_k)$, the Hellinger distance is formally defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2}\,, \tag{3}$$

which is directly related to the Euclidean norm of the difference of the square root vectors:

$$H(P,Q) = \frac{1}{\sqrt{2}} \|\sqrt{P} - \sqrt{Q}\|_2\,. \tag{4}$$

Note that it makes more sense to take the Hellinger distance rather than the Euclidean distance for comparing discrete distributions, as P and Q are unit vectors according to the Hellinger distance ($\sqrt{P}$ and $\sqrt{Q}$ are units vector according to the ℓ_2 norm).

3.1.3. Autoencoder

An autoencoder is employed to represent words in a lower dimensional space. It takes a distribution $\sqrt{P_w}$ as input, encodes it in a more compact representation, and is trained to reconstruct its own input from that representation:

$$||g(f(\sqrt{P_w})) - \sqrt{P_w}||^2\,, \tag{5}$$

where $g(f(\sqrt{P_w}))$ is the output of the network, f is the encoding function which maps distributions in a m-dimension (with $m << |\mathcal{D}|$), and g is the decoding function. $f(\sqrt{P_w})$ is a distributed representation that captures the main factors of variation in the data as the Hellinger PCA does (Bourlard & Kamp, 1988). Here, encoder $f \in \mathbb{R}^{m \times |\mathcal{D}|}$ and decoder $g \in \mathbb{R}^{|\mathcal{D}| \times m}$ are both linear layers.

3.2. Learning to Sum Word Representations

Interesting compositionality properties have been observed from models based on the addition of representations (Mikolov et al., 2013b). An exhaustive comparison of different composition functions has indeed revealed that an additive model performs well on pre-trained word representations (Mitchell & Lapata, 2010). Because our word representations are learnt from linear operations, the inherent structure of these representations is linear. To combine a sequence of words into a common vector space, we then simply apply an element-wise addition of their vector representations. This approach makes sense and works well when the meaning of a text is literally "the sum of its parts". This is usually the case with noun and verb phrase chunks. For example, into phrases such as "the red cat" or "struggle to deal", each word independently has its proper meaning. Distributed representations for such phrase chunks must retain information from the individual words. An objective function is thus defined to learn how to combine the word vector representations, while keeping the maximum information from the original vectors. An operation as simple as a weighted sum will probably fail for sequences where individual words act as operators that modify the meaning of another word, or for multiword expressions. Other more complex functions could be chosen to also include such cases, but we choose to propose a much simpler model (i.e.,

averaging the word representations) to get phrase chunk representations with unsupervised learning. In this paper, we therefore focus on noun and verb phrase chunks.

3.2.1. Additive Model

We define $s = (w_1, \ldots, w_T) \in \mathcal{S}$ a phrase chunk of T words, with $\mathcal{S}$ a set of phrase chunks. By feeding all $\sqrt{P_w}$ into the autoencoder, a representation $\mathrm{x}_w \in \mathbb{R}^m$ of each word $w \in \mathcal{D}$ is obtained:

$$\mathrm{x}_w = f(\sqrt{P_w})\,. \tag{6}$$

By an element-wise addition, a representation of the phrase chunk s can be calculated as:

$$\mathrm{x}_s = \frac{1}{T} \sum_{w_t \in s} \mathrm{x}_{w_t}\,. \tag{7}$$

3.2.2. Training

In predictive-based model, such as the Skip-gram model, the objective is to maximize the likelihood of a word based on other words in the same sequence. Instead, our training is slightly different in the sense that we aim at discriminating whether words are in the phrase chunk or not. An objective function is thus defined to encourage words w_t which appear in the chunk s to give high scores when calculating the dot product between x_{w_t} and x_s. On the other hand, these scores must be low for words $w_i \notin s$ that do not appear in the chunk. We train this problem with a ranking-type cost:

$$\sum_{s \in \mathcal{S}} \sum_{w_t \in s} \sum_{\substack{w_i \in \mathcal{W} \\ w_i \notin s}} \max(0, 1 - \mathrm{x}_s \cdot \mathrm{x}_{w_t} + \mathrm{x}_s \cdot \mathrm{x}_{w_i})\,. \tag{8}$$

Note that due to the large size of $\mathcal{W}$, a negative sampling approach can be used to speed up the training. In Equation 8, the whole dictionary $\mathcal{W}$ is thus replaced by a subset $\mathcal{W}^- \subseteq \mathcal{W}$ with N randomly chosen negative samples $w_i \notin s$. A new set $\mathcal{W}^-$ is randomly picked at each iteration during the training.

3.3. Joint Learning

In contrast with other methods which have subsequently found nice compositionality properties by simple summation, the novelty of our method is the explicit learning of word representations suitable for summation. The system is then designed to force words with similar context to be close in a m-dimensional space, while these dimensions are learnt to be combined with other related words. This joint learning is illustrated in Figure 1. The whole system is trained by minimizing both objective functions (5) and (8) over the training data using stochastic gradient descent.

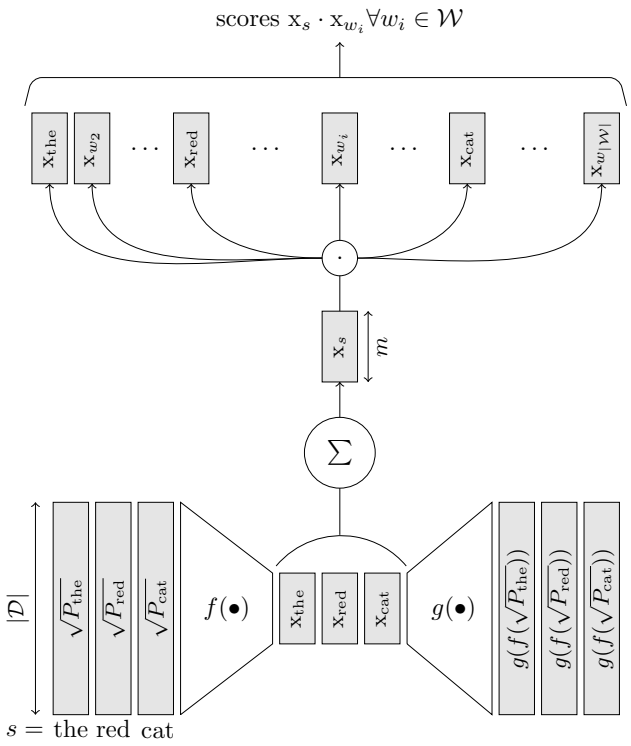

Figure 1. Architecture for the joint learning of word representations and their summation. Considering the noun phrase $s =$ `the red cat`, each word $w_t \in s$ is represented as the square root of its co-occurrence probability distribution $\sqrt{P(w_t)}$. These are the inputs given to an autoencoder which encodes them in a lower dimension $\mathrm{x}_{w_t} \in \mathbb{R}^m$. These new representations are then given to a decoder which is trained to reconstruct the initial inputs. This is the first objective function. The second objective is to keep information when words are summed. All x_{w_t} are summed together to represent s in the same space as w_t. A dot product between the phrase representation x_s and all the other word representations from the dictionary $\mathcal{W}$ is calculated. These scores are trained to be high for words that appear in s and low for the others.

4. Experiments

4.1. Datasets

4.1.1. Building Word Representation over Large Corpora

Our English corpus is composed of the entire English Wikipedia[1] (where all MediaWiki markups have been removed). We consider lower case words to limit the number of words in the dictionary. Additionally, all occurrences of sequences of numbers within a word are replaced with the string "NUMBER". The resulting text is tokenized using the Stanford tokenizer[2]. The data set contains about 1.6

[1] Available at `http://download.wikimedia.org`. We took the January 2014 version.

[2] Available at `http://nlp.stanford.edu/software/tokenizer.shtml`

billion words. As dictionary $\mathcal{W}$, we consider all the words within our corpus which appear at least one hundred times. This results in a 191,268 words dictionary. Only the 10,000 most frequent words within this dictionary were used as context words $\mathcal{D}$ to calculate the word co-occurrence probabilities. A symmetric context window of ten words around each word $w \in \mathcal{W}$ is used to obtain the multinomial distribution P_w. We chose to encode words in a 100-dimensional vector.

4.1.2. Summing Words for Phrase Representation

To learn the summation of words that appear frequently together, we choose to consider only the noun and verb phrase chunks to build $\mathcal{S}$. We extract these chunks with a phrase chunking approach by using the SENNA software[3]. By retaining only the phrase chunks appearing at least ten times, this results in 1,823,259 noun phrase chunks and 255,232 verb phrase chunks, for a total of 2,078,491 phrase chunks. We divided this set of phrases into three sets: 1,000 phrases for validation, 5,000 phrases for testing, and the rest for training (2,072,491 phrases). An unsupervised framework requires a large amount of data. Because our primary focus is to provide good word representations, validation and testing sets are intentionally kept small to retain as much phrases as possible in the training set.

4.2. Other Methods

We compare our distributed representations with other available models for computing vector representations of words: (1) the GloVe model which is also based on co-occurrence statistics of corpora (Pennington et al., 2014)[4], (2) the continuous bag-of-words (CBOW) and the skip-gram (SG) architectures which learn representations from prediction-based models (Mikolov et al., 2013b)[5]. The same corpus and dictionary $\mathcal{W}$ as the ones described in Section 4.1.1 are used to train 100-dimensional word vector representations. We use a symmetric context window of ten words, and the default values set by the authors for the other hyperparameters. To see the improvement compared to a standalone SVD, we generate word representations with a truncated SVD of the matrix X, where each row of X is a distribution $\sqrt{P_w}$, $X = \left(\sqrt{P_{w_1}}, \sqrt{P_{w_2}}, \ldots, \sqrt{P_{w_{|\mathcal{W}|}}}\right)^T \in \mathbb{R}^{|\mathcal{W}| \times |\mathcal{D}|}$.

[3]Available at http://ml.nec-labs.com/senna/

[4]Code available at http://www-nlp.stanford.edu/software/glove.tar.gz.

[5]Code available at http://word2vec.googlecode.com/svn/trunk/.

4.3. Evaluating Word Representations

The first objective of the model is to learn distributed representations which capture both syntactic and semantic informations about words. To evaluate the quality of these representations, we used both analogy and similarity tasks.

4.3.1. Word Analogies

The word analogy task consists of questions like, "*a* is to *b* as *c* is to ?". It was introduced in Mikolov et al. (2013a) and contains 19,544 such questions, divided into a semantic subset and a syntactic subset. The 8,869 semantic questions are analogies about places, like "*Bern* is to *Switzerland* as *Paris* is to ?", or family relationship, like "*uncle* is to *aunt* as *boy* is to ?". The 10,675 syntactic questions are grammatical analogies, involving plural and adjectives forms, superlatives, verb tenses, etc. To correctly answer the question, the model should uniquely identify the missing term, with only an exact correspondence counted as a correct match.

4.3.2. Word Similarities

We also evaluate our model on a variety of word similarity tasks. These include the WordSimilarity-353 Test Collection (WS-353) (Finkelstein et al., 2001), the Rubenstein and Goodenough dataset (RG-65) (Rubenstein & Goodenough, 1965), and the Stanford Rare Word (RW) (Luong et al., 2013). They all contain sets of English word pairs along with human-assigned similarity judgements. WS-353 and RG-65 datasets contain 353 and 65 word pairs respectively. Those are relatively common word pairs, like *computer:internet* or *football:tennis*. The RW dataset differs from these two datasets, since it contains 2,034 pairs where one of the word is rare or morphologically complex, such as *brigadier:general* or *cognizance:knowing*.

4.3.3. Results

	WS	RG	RW	SYN.	SEM.
CBOW	0.57	0.47	0.32	53.5	22.7
Skip-gram	**0.62**	0.49	**0.39**	66.7	53.8
GloVe	0.56	**0.50**	0.36	**79.7**	**75.0**
SVD	0.43	0.39	0.27	52.3	34.1
Our model	**0.62**	0.49	**0.39**	69.4	43.0

Table 1. Word representations evaluation on both similarity and analogy tasks. Comparison of performance across all models with 100-dimensional word vector representations. For all models, a symmetric context window of ten words is used. Spearman rank correlation is reported on word similarity tasks. Accuracy is reported on word analogy tasks.

Results reported in Table 1 show that our model gives similar results than other state-of-the-art methods on word similarity tasks. However, there is a significant performance boost between the low-rank approximation of X with a SVD and this same approximation with our joint model. This shows that combining a count-based model with a predictive-based approach helps for generating better word representations. Performance on word analogy tasks show that our joint model competes with others on the syntactic questions, but that it gives a lower accuracy on semantic questions. One possible explanation is that less common words are involved in semantic questions compared to syntactic questions. Among the four words that make a semantic question, one of them is, in average, the 34328^{th} most frequent word in $\mathcal{W}$, while it is the 20819^{th} for a syntactic question. Compared to other methods which take the whole dictionary $\mathcal{W}$ as context dictionary, we consider only a small subset of it ($\mathcal{D}$ contains only the 10000 most frequent words of $\mathcal{W}$). A larger context dictionary would certainly help to improve performance on this task[6].

4.4. Evaluating Phrase Representations

As a second objective, we aim at learning to sum word representations to generate phrase representations while keeping the original information coming from the words. We thus introduce a novel task to evaluate the phrase representations.

4.4.1. Description of the Task

As dataset, we use the collection of test phrases described in Section 4.1.2. It contains 5000 phrases (noun phrases and verb phrases) extracted from Wikipedia with a chunking approach. Amoung them, 2244, 2030 and 547 are, respectively, composed of two, three and four words. The remaining 179 are composed of at least five words with a maximum of eight words. For a given phrase $s = (w_1, \dots, w_T) \in \mathcal{S}$ of T words, the objective is to retrieve the T words from its distributed representation x_s. Scores between the phrase s and all the possible words $w_i \in \mathcal{W}$ are calculated using the dot product between their distributed representations $\mathrm{x}_s \cdot \mathrm{x}_{w_i}$, as illustrated in Figure 1. The top T scores are considered as the words composing the phrase s.

4.4.2. Results

To evaluate whether words making a given phrase can be retrieved from the distributed phrase representation, we use Recall @K, which measures the fraction of times a correct word was found among the top K results. K is proportional to the number of words per phrase, e.g. for a 3-word phrase with a Recall@5, the correct words are found amoung the top 15 results. Higher Recall @K means better retrieval performance. Since we care most about the top-ranked retrieved results, the Recall @K with small K are more important.

[6]It has not been explored due to limitations in hardware resources. It would be easily computable with a cluster of CPU.

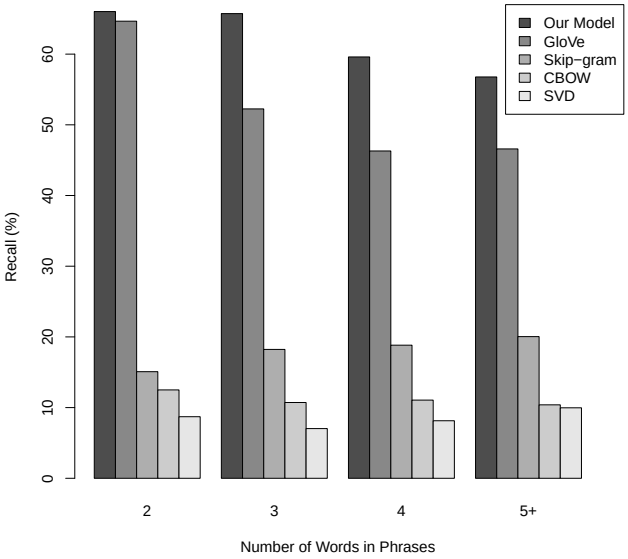

Figure 2. Recall@1 based on the number of words per phrases. Comparison of performance across all models with 100-dimensional word vector representations.

	R@1	R@5	R@10
CBOW	11.33	29.56	38.46
Skip-gram	7.96	22.26	30.04
GloVe	54.97	79.97	86.54
SVD	17.42	32.87	40.72
Our model	**64.22**	**91.72**	**95.85**

Table 2. Phrase representations evaluation. Comparison of performance across all models with 100-dimensional phrase vector representations on word retrieval. R@K is Recall@K, with $K = \{1, 5, 10\}$.

Results reported in Table 2 show that our distributed word representations can be averaged together to produce meaningful phrase representations, since the words are retrieved with a high recall. Our model significantly outperforms others methods on this task. In Figure 2, a more detailed analysis of results reveals that the GloVe model competes with ours for the 2-word phrases. However GloVe's representations cannot maintain this performance for longer phrases. It is probably not too surprising as this model is trained using ratios of co-occurrence probabilities for two

QUERY PHRASES	NEAREST PHRASES	
	ENCODING FUNCTION f	AVERAGING WORDS
AMERICAN AIRLINES	BRANIFF AIRLINES	AMERICAN AIRWAYS
	ALOHA AIRLINES	PAN AMERICAN AIRLINES
	BRANIFF AIRWAYS	AMERICAN EAGLE AIRLINES
	JETBLUE AIRWAYS	NORTH AMERICAN AIRLINES
	BRANIFF INTERNATIONAL AIRWAYS	AMERICAN OVERSEAS AIRLINES
CHICAGO BULLS	DENVER NUGGETS	CHICAGO COLTS
	SEATTLE SUPERSONICS	CHICAGO HORNETS
	CLEVELAND CAVALIERS	CHICAGO STAGS
	BOSTON CELTICS	BUFFALO BULLS
	DALLAS MAVERICKS	CHICAGO CARDINALS
HOME PLATE	RIGHT FIELDER	THE HOME PLATE UMPIRE
	CENTER FIELDERS	THE HOME PLATE AREA
	THE OUTFIELD FENCE	THE HOME LEG
	LEADOFF BATTER	THE BALL HOME
	THE INFIELD	THE DIAMOND STATE BASE BALL CLUB
PRESIDENT OF THE UNITED STATES	PRESIDENT COOLIDGE	THE UNITED STATES PRESIDENT
	PRESIDENT EISENHOWER	THE UNITED STATES PRESIDENCY
	U.S. PRESIDENT DWIGHT EISENHOWER	THE FIRST UNITED STATES SECRETARY
	PRESIDENT TRUMAN	THE UNITED STATES MINISTER
	PRESIDENT REAGAN	THE FIRST UNITED STATES SENATOR

Table 3. Examples of phrases and five of their ten nearest phrases from the collection of phrases. Representations for the collection of phrases have been computed by averaging the word representations. Query phrase representations are inferred using the two different alternatives: (1) with the encoding function f using counts from a symmetric window of ten context words around the query phrase, (2) by averaging the representations of the words that compose the query phrase. All distributed representations are 100-dimensional vectors.

target words. Consequently, it well learns linear substructures for pairs of words, which probably also explains its good performance on word analogy tasks. In contrast, our joint model can learn more complex substructures which make possible the aggregation of multiple words within a low-dimensional vector space.

4.5. Inferring New Phrase Representations

Representations for new phrases can thus be generated by simply averaging its word representations, assuming that the words are in the dictionary $\mathcal{W}$. Consider that the dictionary $\mathcal{W}^n$ tends to grow exponentially with n, it gives a nice framework to produce the huge variety of possible sequence of n words in a timely and efficient manner with low memory consumption, unlike other methods. Relying on word co-occurrence statistics to represent words in vector space also provides a framework to easily generate representations for unseen words. This is another advantage compared to methods focused on learning distributed word representations (such as CBOW, Skip-gram and GloVe models), where the whole system needs to be trained again to learn representations for these new constituents. To infer a representation for a new word w_{new}, one only needs to count its context words over a large corpus of text to build the distribution $\sqrt{P_{w_{\text{new}}}}$. This nice feature can be extrapolated to phrases, which gives another alternative for generating phrase representations. Table 3 presents some examples of phrases, where we use both alternatives to compute their distributed representations. It can be seen that both alternatives give distinct representations. For instance, by using the encoding function f, our model infers a representation for the entity *Chicago Bulls* which is close to other NBA teams, like the *Denver Nuggets* or the *Seattle Supersonics*. By averaging the representations of both words *Chicago* and *Bulls*, our model infers a representation which is close to other Chicago's sport teams. Both representations are meaningful, but they carry different information. Relying on co-occurrence statistics gives entities that occur in a similar context, while the summation tries to find entities containing the maximum amount of similar information. This also works with longer phrases, such as *President of the United States*. The first alternative gives men who served as president, when the second gives related positions.

5. Conclusion

We introduce a model that combines both count-based methods and predictive-based methods for generating distributed representations of words and phrases. Using a chunking approach, a collection of noun phrases and verb phrases is extracted from Wikipedia. For a given n-word phrase, we train our model to generate a low-dimensional representation for each word based on its co-occurrence probability distribution. These n representations are averaged together to generate a distributed phrase representation in the same semantic space. Thanks to an autoencoder approach, we can simultaneously train the model to retrieve the original n words from the phrase representation, and therefore learn complex linear substructures. When compared to state-of-the-art methods on some classical word evaluation tasks, the competitive results show that our joint model produces meaningful word representations. Performance on a novel task for evaluating phrase representations confirm the ability of our model to learn complex substructures, which make possible the aggregation of multiple words within a low-dimensional vector space. Better still, inference of new phrase representations is also easily feasible when relying on counts. Some quantitative examples demonstrate that both alternatives can give different but meaningful information about phrases. The word representations and the collection of phrases used in these experiments are available online, here: `http://www.lebret.ch/words/`.

Acknowledgements

This work was supported by the HASLER foundation through the grant "Information and Communication Technology for a Better World 2020" (SmartWorld).

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3th International Conference on Learning Representations (ICLR)*, 2015.

Baroni, M. and Zamparelli, R. Nouns are Vectors, Adjectives are Matrices: Representing Adjective-Noun Constructions in Semantic Space. In *Proceedings of the EMNLP*, pp. 1183–1193, 2010.

Bengio, Y. Neural net language models. *Scholarpedia*, 3 (1):3881, 2008.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

Blacoe, W. and Lapata, M. A Comparison of Vector-based Representations for Semantic Composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pp. 546–556. Association for Computational Linguistics, 2012.

Bourlard, H. and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

Bullinaria, J. A. and Levy, J. P. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526, 2007.

Clark, S., Coecke, B., and Sadrzadeh, M. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pp. 133–140, 2008.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural Language Processing (Almost) from Scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., and Ruppin, E. Placing Search in Context: The Concept Revisited. In *Proceedings of the 10th international conference on World Wide Web*, pp. 406–414. ACM, 2001.

Firth, J. R. A Synopsis of Linguistic Theory 1930-55. 1957.

Grefenstette, E., Dinu, G., Zhang, Y., Sadrzadeh, M., and Baroni, M. *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, chapter Multi-Step Regression Learning for Compositional Distributional Semantics, pp. 131–142. Association for Computational Linguistics, 2013.

Harris, Z. S. Distributional structure. *Word*, 1954.

Humboldt, W. *Über die Verschiedenheit des menschlichen Sprachbaues: Und ihren Einfluss auf die geistige Entwickelung des Menschengeschlechts*. Druckerei der Königlichen Akademie der Wissenschaften, 1836.

Lebret, R. and Collobert, R. Word Embeddings through Hellinger PCA. In *Proceedings of the 14th Conference*

of the European Chapter of the Association for Computational Linguistics, pp. 482–490. Association for Computational Linguistics, April 2014.

Lebret, R., Pinheiro, P. H. O., and Collobert, R. Phrase-based image captioning. In *Proceedings of the 32st International Conference on Machine Learning (ICML)*, 2015.

Legrand, J. and Collobert, R. Joint rnn-based greedy parsing and word composition. In *Proceedings of the 3th International Conference on Learning Representations (ICLR)*, 2015.

Levy, O. and Goldberg, Y. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems*, pp. 2177–2185. 2014.

Lund, K. and Burgess, C. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, 1996.

Luong, M., Socher, R., and Manning, C. D. Better Word Representations with Recursive Neural Networks for Morphology. In *CoNLL*, 2013.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. *In Proceedings of Workshop at International Conference on Learning Representations (ICLR 2013)*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pp. 3111–3119. 2013b.

Mitchell, J. and Lapata, M. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

Mnih, A. and Kavukcuoglu, K. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pp. 2265–2273, 2013.

Patel, M., Bullinaria, J. A., and Levy, J. P. Extracting Semantic Representations from Large Text Corpora. In *4th Neural Computation and Psychology Workshop, London, 9–11 April 1997*, pp. 199–212. Springer, 1998.

Pennington, J., Socher, R., and Manning, C. D. GloVe: Global Vectors for Word Representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12, 2014.

Rubenstein, H. and Goodenough, J. B. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10): 627–633, 1965.

Socher, R., Huval, B., Manning, C., and Ng, A. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201–1211. Association for Computational Linguistics, 2012.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y, and Potts, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1631, pp. 1642. Association for Computational Linguistics, 2013.

Turney, P. and Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.