attr2vec: Jointly Learning Word and Contextual Attribute Embeddings with Factorization Machines

Fabio Petroni
Thomson Reuters
Corporate Research & Development
30 South Colonnade, London, UK
fabio.petroni@tr.com

Vassilis Plachouras*
Facebook
1 Rathbone Square, London, UK
vplachouras@fb.com

Timothy Nugent
Thomson Reuters
Corporate Research & Development
30 South Colonnade, London, UK
tim.nugent@tr.com

Jochen L. Leidner
Thomson Reuters
Corporate Research & Development
30 South Colonnade, London, UK
jochen.leidner@tr.com

Abstract

The widespread use of word embeddings is associated with the recent successes of many natural language processing (NLP) systems. The key approach of popular models such as word2vec and GloVe is to learn dense vector representations from the context of words. More recently, other approaches have been proposed that incorporate different types of contextual information, including topics, dependency relations, n-grams, and sentiment. However, these models typically integrate only limited additional contextual information, and often in *ad hoc* ways.

In this work, we introduce attr2vec, a novel framework for jointly learning embeddings for words and contextual attributes based on factorization machines. We perform experiments with different types of contextual information. Our experimental results on a text classification task demonstrate that using attr2vec to jointly learn embeddings for words and Part-of-Speech (POS) tags improves results compared to learning the embeddings independently. Moreover, we use attr2vec to train dependency-based embeddings and we show that they exhibit higher similarity between functionally related words compared to traditional approaches.

1 Introduction

Neural network-based methods have been successful in advancing the state-of-the-art in a wide range of NLP tasks, such as dependency parsing (Chen and Manning, 2014), sentence classification (Kim, 2014), machine translation (Sutskever et al., 2014; Luong and Manning, 2016), and information retrieval (Zhang et al., 2017). In all these approaches, vectorial distributed word representations, known as *word embeddings*, have become a fundamental building block. The use of word embeddings is considered a "secret sauce" for contributing to the success of many of these algorithms in recent years (Luong et al., 2013). Popular models for learning such word embeddings include word2vec (Mikolov et al., 2013a,b,c), GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2017; Joulin et al., 2017).

The main idea behind these techniques is to represent a word by means of its context. The most popular forms of context are neighboring words in a window of text (Mikolov et al., 2013b; Pennington et al., 2014), though examples of additional contextual information might also include document topics (Li et al., 2016), dependency relations (Levy and Goldberg, 2014), morphemes (Luong et al., 2013), n-grams (Bojanowski et al., 2017), and sentiment (Tang et al., 2014). The embedding idea was originally devised to help overcome problems associated with the high dimensionality of sparse vector representations of words, particularly in the case of neural network modeling, though embeddings have since been used in a variety of machine learning approaches.

However, existing models generally exploit just a small portion of the available contextual information, and they tend to do so in *ad hoc* ways. The main purpose of context in these models is to shape the word vector space (that is, to associate a representation to the word), but contextual information is not usually represented in this space. For instance, Li and Jurafsky (2015) used document topics to derive multiple vectors for the same word, each capturing a different sense, but

*work conducted whilst the author was at Thomson Reuters.

Proceedings of NAACL-HLT 2018, pages 453–462
New Orleans, Louisiana, June 1 - 6, 2018.

their method does not represent topics in the vector space, that is, it does not generate topic vectors. Such contextual representations, jointly learned with the word representations, could potentially be useful for multiple tasks. For instance, pre-trained contextual vectors could be used as additional features, together with pre-trained word vectors, to improve the performance of existing models.

In this paper, we propose attr2vec, a novel framework for learning word embedding models that jointly associate distributed representations with words and with generic contextual attributes. attr2vec is inspired by the GloVe approach of Pennington et al. (2014) and can mimic it when no additional contextual attribute is considered. In contrast with GloVe, attr2vec uses Factorization Machines (FMs) (Rendle et al., 2011; Rendle, 2012). FMs are a generalization of matrix factorization approaches, such as GloVe, and can combine different generic feature types, even when the input data is sparse. Moreover, FMs do not consider input features as independent but model their interaction by factorizing their latent representation in pairwise fashion.

Here, we conduct an experimental study to assess whether the proposed embedding model can lead to better performance for a text classification task on the *Reuters-21578* dataset, using trained vectors as input to a convolutional neural network. The results show that jointly learned word and Part-of-Speech (POS) embeddings with attr2vec can achieve higher F1 and precision scores compared to embeddings learned independently. Moreover, we use attr2vec to train dependency-based word embeddings and show, using the publicly available *WordSim353* dataset, that such embeddings yield more *functional* similarities than embeddings trained using a linear bag-of-word approach (such as GloVe). We also performed a qualitative analysis that provides insights on how contextual attributes affects the distribution of words in the vector space.

Summing up, the main contributions of our work are the following:

- we extend the GloVe model to consider additional contextual information. To the best of our knowledge, this is the first work to present a general model able to jointly train dense vector representations for word and multiple arbitrary contextual attributes;
- we define a novel loss function based on factorization machines to jointly learn word and contextual attribute embeddings;
- we show how to model the input data and compute co-occurrence statistics using either a linear bag-of-word approach or syntactic dependency relations.

We provide the source code for the attr2vec model at `https://github.com/thomsonreuters/attr2vec`.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work and Section 3 introduces the attr2vec model. In Section 4, we present the experimental results, and close this paper with some concluding remarks in Section 5.

2 Related Work

We have already introduced some of the main related approaches including word2vec and GloVe. Essentially, the GloVe model (Pennington et al., 2014) derives word representations by factorizing the word co-occurrence count matrix. The skip-gram and continuous bag-of-words (CBOW) models of Mikolov et al. (2013a), instead, build the vector space by trying to predict a word given its neighbouring words. Mnih and Kavukcuoglu (2013) proposed a closely related model that works in the opposite way, trying to predict neighbouring words given a word. Facebook's fastText model (Bojanowski et al., 2017; Joulin et al., 2017) augments word embeddings with subword-level information using character n-grams. Other examples of word embedding models include the work of Levy et al. (2014), where an explicit word vector space representation is derived using a PPMI metric, and WordRank of Ji et al. (2016), which learns word representations by adopting a ranking-based loss function. However, none of these models includes any contextual information beyond the neighbouring words.

Several forms of contextual information have been successfully integrated into word embedding models. For instance, Luong et al. (2013); Cotterell and Schütze (2015); Bhatia et al. (2016) capture morphological information into word representations; Bojanowski et al. (2017); Wieting et al. (2016) include character n-grams in their embedding model; Tang et al. (2014) learn sentiment-specific word embeddings by integrating sentiment information in the loss function; Li et al.

(2016) combine word embedding and topic modeling to jointly learn a representation for topics and words. In addition, several works in recent years focused on learning separate embeddings for multiple senses of a word (Neelakantan et al., 2015; Iacobacci et al., 2015; Pilehvar and Collier, 2016). However, all these techniques target a particular type of context. Our attr2vec model differs in that it can jointly represent generic contextual attributes and words in the embedding model. To do so, it makes use of factorization machines (Rendle, 2012), which have been successfully used to exploit contextual information in relation extraction tasks (Petroni et al., 2015) and recommender systems (Rendle et al., 2011)

It is well known that contextual information can improve performance of a wide range of NLP tasks, such as machine translation (Koehn and Hoang, 2007; García-Martínez et al., 2017), named entity typing (Corro et al., 2015) or sentiment analysis (Weichselbraun et al., 2013). In addition, (Melamud et al., 2016) observed that different contextual attributes work well for different tasks and that simple concatenation of embeddings, learned independently with different models, can yield further performance gains. Our attr2vec model can jointly learn embeddings for words and contextual attributes, and we show (see Section 4) that using such jointly learned embeddings yields to better performance on a text classification task compared to embeddings learned independently.

3 The attr2vec model

This section presents the attr2vec model. We first describe how we model the input data in terms of a feature matrix and a target vector (Section 3.1) and then how to factorize those using a factorization machines-based formulation (Section 3.3) to obtain word and contextual attribute embeddings.

3.1 Modeling input data

We consider as input a large corpus of text. Let the vocabulary of considered words be denoted by $W = \{w_1, ..., w_{|W|}\}$ and the set of all contextual variables denoted by $C = \{c_1, ..., c_{|C|}\}$. We denote $V = W \cup C$ the set of all considered words and contextual variables. In the rest of the paper we will refer to the elements of V as *variables*. We model the input data in terms of a target vector $Y \in \mathbb{R}^m$ and feature matrix $X \in \mathbb{R}^{m \times n}$, in which each row $x_i \in \mathbb{R}^n$ corresponds to a feature vector and there are as many columns as the number of variables (i.e., $|V| = n$). We group columns according to the type of the variables; e.g., there are word columns and a group of columns for each type of contextual information considered. Each target value $y_i \in Y$ represents the number of times the feature vector x_i has been observed in the input (i.e., the co-occurrence count). We consider a two-fold way to compute the co-occurrence count of a feature vector x_i: (1) *linear bag-of-words* and (2) *dependency-based*.

Linear Bag-of-Words The approach used in Pennington et al. (2014) is to compute the co-occurrence count using a linear bag-of-words assumption. The idea is to use a window of size k around the target word w, and considering the k words before and the k words after w to compute co-occurrence statistics[1]. Note that a small window size may miss some information, while a large window size might result in the algorithm capturing accidental pairs. A decay factor is commonly used to weight the contribution of an observation to the total co-occurrence count according to the distance to the target word. Here we consider a fractional decay, where the importance of a word is assumed to be inversely proportional to its distance from the target word.

To build the feature matrix X we set the values of the variables associated with the words pair and with each contextual variable observed in correspondence with that pair to 1. Note that there could be multiple rows in X referring to the same pair of words but associated with different contextual variables. When contextual variables can assume multiple values for a single observation (e.g., a document with multiple topics) we evenly distribute the unitary weight across all variables (e.g., across all document topics). The target values represent the number of times the corresponding combination of features (i.e., the pair of word and the contextual variables) has been observed in the input corpus, weighting each contribution with the fractional decay factor described above.

An example is shown in Figure 1. The first row in the figure corresponds to the observation of the pair of words *brothers* and *lehman* in a text window, with POS tags *nnp* and *nnps*, respectively, referring to the named entity *Lehman*

[1] In this paper we focus on symmetric windows, however the same model can be extended to asymmetric windows.

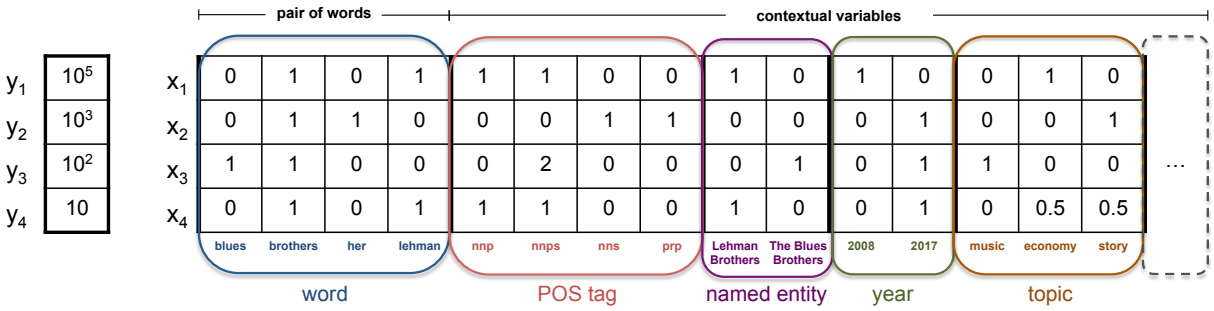

Figure 1: Example for representing input data with attr2vec using a linear window of text approach to compute co-occurrence count. Rows in X and Y are aligned: the row $y_i \in Y$ (a scalar) correspond to the row $x_i \in X$ (a feature vector); y_i represents the frequency of the particular combination of variables described by x_i in the corpus.

Brothers, from documents published in *2008* with topic *economy*. Such a combination of features (i.e., *brothers-lehman-nnp-nnps-Lehman Brothers-2008-economy* has been observed in the input 10^5 times (i.e., $y = 10^5$). The fourth row of Figure 1 conveys the information that the same pair of words (i.e., *brothers-lehman*) has been also observed 10 times in the input associated with different contextual information, in particular in documents published in *2017* with multi-topic *economy* and *story*. When contextual information is not considered (i.e., $c = \emptyset$) y is equal to the co-occurrence count of the pair of words, as in Pennington et al. (2014).

Dependency-Based Our attr2vec model can learn dependency-based embedding as well. In order to do so we adopted a similar strategy of Levy and Goldberg (2014). The main idea is to parse each sentence in the input corpus and to use the dependency tree to derive the co-occurrence count. In particular, for a target word w with modifiers $m_1, ..., m_o$ and a head h, we considered the dependency labels $(m_1, lbl_1), ..., (m_o, lbl_o), (h, lbl_h^{-1})$, where lbl is the type of the dependency relation between the head and the modifier and lbl^{-1} is used to mark the inverse relation. Moreover, edges that include a preposition are "collapsed" by connecting the head and the object of the proposition, and subsuming the preposition itself into the dependency label (see the example in the top part of Figure 2).

The feature matrix X in this case consists of two group of columns, one for the words and one for dependency labels, plus one group of columns for each additional contextual information considered. The co-occurrence count is driven by the dependency tree: each target value represents the number of times the corresponding word and dependency label (and all considered additional contextual information) appear in the dependency trees representing the sentences in the input corpus.

An example is shown in Figure 2. The first row in the matrix corresponds to the observation of the word *ganymede* connected to the word *discovered* through the inverse relation $dobj$ in the dependency tree (i.e., $discovered/dobj^{-1}$). Notice that using this approach it is possible to capture relevant relations between words "far apart" in the text including long-distance dependencies (e.g., *telescope* is not in the window of text around *discovered* for $k = 3$), and also filter out accidental neighbouring words (e.g., *his* is in the window of text of *discovered* for $k = 3$). An additional advantage of this approach with respect to a linear bag-of-words solution is that each observation is typed, indicating, for instance, that *Ganymede* is the object of *discovered* and *Galileo* is the subject.

3.2 GloVe factorization model

Before introducing the attr2vec model we briefly describe the GloVe factorization approach (Pennington et al., 2014). In particular, GloVe employs a matrix factorization model using as input the word-word co-occurrence count. In our example of Figure 1 this corresponds to considering in input just the first group of columns (i.e., the pair of words).

Starting from the observation that the ratio of co-occurrence probabilities is more appropriate for learning word representations as opposed to the probabilities of the words themselves, Pennington et al. propose the following weighted least

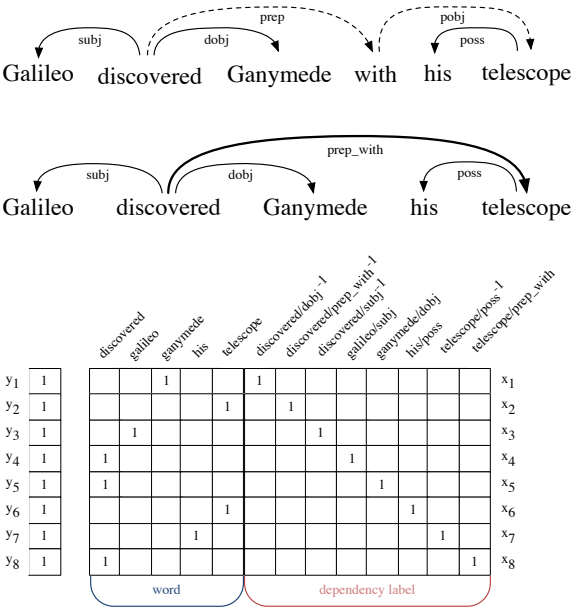

Figure 2: Example for representing input data with attr2vec using a dependency-based approach to compute co-occurrence count. **Top:** preposition relations are collapsed into single arcs, making telescope a direct modifier of discovered. **Bottom:** the attr2vec data matrix. Note that the Y vector contains all 1's since it refer to the modeling of a single sentence; in a real scenario, each row y_i of vector Y will contain a higher value (the frequency count of x_i in the corpus)

squares objective function:

$$J = \sum_{k=1}^{N} f(y_k)\Big(s(x_k) - \log(y_k)\Big)^2 \qquad (1)$$

where $x_k = (w_i, w_j)$ refers to the k-th pair of words in input with co-occurrence count y_k, and $s(x)$ is the score associated with the pair, computed as follows:

$$s(x) = b_{w_i} + b_{w_j} + \boldsymbol{f}_{w_i}^T \boldsymbol{f}_{w_j} \qquad (2)$$

where b_{w_i} and b_{w_j} are the biases, and $\boldsymbol{f}_{w_i}$ and $\boldsymbol{f}_{w_j}$ are the latent factor vectors associated with w_i and w_j, respectively.

The function $f(y)$ is used to reduce the importance of pairs of words that co-occur rarely, and is defined as follows:

$$f(y) = \begin{cases} (y/y_{max})^{\alpha} & \text{if } y < y_{max} \\ 1 & \text{otherwise} \end{cases} \qquad (3)$$

where y_{max} and α are hyperparameters of the model.

3.3 attr2vec factorization model

The model we propose, attr2vec, employs a matrix factorization model based on factorization machines. In particular, we associate with each variable $v \in V$ a *bias term* $b_v \in \mathbb{R}$ and a *latent factor vector* $\boldsymbol{f}_v \in \mathbb{R}^d$, where the *dimensionality* d of the latent factor space is a hyperparameter of the model. For each input feature vector $x \in X$, we denote by x_v the value of variable $v \in V$ in the corresponding row of the features data matrix.

We employ a weighted least squares model that is based of the formulation of Pennington et al. (2014) (Equation 1). In contrast to GloVe, we define a novel score $s(x)$ that takes into account both words and contextual attributes, computed as follows:

$$s(x) = \sum_{v \in V} x_v b_v + \sum_{v_1 \in V} \sum_{v_2 \in V \setminus \{v_1\}} x_{v_1} x_{v_2} \boldsymbol{f}_{v_1}^T \boldsymbol{f}_{v_2} \qquad (4)$$

Here, the bias terms b_v model the contribution of each individual variable to the final score, whereas the latent factor vectors $\boldsymbol{f}_v$ model the contribution of all pairwise interactions between variables. Rendle (2012) has shown that score computation is fast, since $s(x)$ can be computed in time linear to both the number of nonzero entries in x and the dimensionality d. Each latent factor vector can be interpreted as a low-dimensional representation of the corresponding variable, both for variables that refer to words and for variables that refer to contextual information. Note that, when contextual information is not considered the formulation of our factorization model is equivalent to the formulation of Pennington et al. (2014).

The model parameters $\Theta = \{b_v, \boldsymbol{f}_v | v \in V\}$ are estimated by minimizing J, for instance, through stochastic gradient descent. Each $\boldsymbol{f}_v$ can be interpret as a dense vector representation of variable $v \in V$.

4 Experiments

We conducted an experimental study on real-world data to compare our attr2vec model with other state-of-the-art approaches.

4.1 Dataset and Baseline

For a training corpus to learn embeddings, we used the Reuters News Archive, in particular, the collection of all news stories published by the Reuters News Agency from 2003 to 2017. We

embedding	*input*	*logistic regression*	*convolutional neural network*	
			static	*non-static*
random	$\vec{w_r}$	70.5 (69.3)	75.7 (77.5)	74.4 (76.2)
random	$\vec{w_r} \frown \vec{p_r}$	74.0 (73.9)	77.9 (79.7)	77.8 (79.8)
GloVe	$\vec{w_i}$	77.5 (77.5)	79.7 (81.5)	82.7 (84.3)
GloVe	$\vec{w_i} \frown \vec{p_r}$	80.2 (85.4)	82.5 (84.1)	84.5 (86.1)
GloVe	$\vec{w_i} \frown \vec{p_i}$	79.3 (83.3)	84.3 (85.8)	84.9 (86.4)
attr2vec	$\vec{w_j}$	77.5 (77.3)	80.6 (82.3)	82.8 (84.5)
attr2vec	$\vec{w_j} \frown \vec{p_j}$	80.1 (83.1)	84.9 (86.1)	**85.5** (**86.8**)

Table 1: Average F1 score (and precision in parentheses) for topic prediction on the *Reuters-21578* dataset. $\vec{w} \frown \vec{p}$ indicates the concatenation of word and POS tag vectors. $\vec{w_r}$ refer to randomly initialized word vectors and $\vec{p_r}$ to randomly initialized POS tag vector. $\vec{w_i}$ and $\vec{p_i}$ respectively refer to vectors independently trained with the GloVe model for words and POS tags; $\vec{w_j}$ and $\vec{p_j}$ respectively refer to vectors jointly trained with attr2vec for words and POS tags.

first applied a heuristic filtering approach to exclude non-textual documents, resulting in a collection of $\sim 8M$ news articles ($\sim 3B$ tokens). We then performed tokenization, part-of-speech tagging, and syntactic dependency parsing on the corpus using NLP4J[2] (Choi et al., 2015; Choi, 2016). The POS tagger achieves an accuracy score of 97.64% (Choi, 2016), the dependency parser achieve a label accuracy score of 94.94% (Choi et al., 2015). As a baseline we considered 200-dimensional GloVe vectors trained on the corpus using the code and hyperparameters of Pennington et al. (2014). In particular, we used $y_{max} = 100$ and $\alpha = 3/4$ for all our experiments.

4.2 Topic Classification Experiment

Experimental Setup For this experiment we trained 200-dimensional attr2vec vectors using part-of-speech tags as additional contextual information and a linear bag-of-words approach - i.e., each row in the feature matrix consists of a pair of words and the corresponding pair of POS tags (see the first two groups of columns for the example in Figure 1). We used the same hyperparameters as in GloVe. To make a fair comparison we trained two independent GloVe models, one to obtain word vectors ($\vec{w_i}$) and one to obtain POS tag vectors ($\vec{p_i}$). The latter model is trained by substituting each word in the corpus with the corresponding POS tag. Note that our attr2vec model can jointly learn a representation for both words ($\vec{w_j}$) and POS tags ($\vec{p_j}$). As a baseline we also considered randomly initialized vectors for words ($\vec{w_r}$) and POS tags ($\vec{p_r}$). To train attr2vec we used a modified version[3] of tffm (Trofimov and Novikov, 2016), an open-source TensorFlow implementation of Factorization Machines.

To evaluate the performance of the attr2vec model, we used the trained vectors as input for a convolutional neural network (CNN). We used the CNN architecture described by Kim (2014), in particular a modified version of the TensorFlow implementation in Britz (2015), where we add support for pre-trained embeddings. As hyperparameters, we used a batch size of 128 training samples, no dropout, one layer, filter windows of 3, 4, 5 with 100 feature maps each. We trained using the Adam optimizer and a learning rate of 0.001 and let the models train for 100 epochs (an epoch is an iteration over all the training points). We executed three independent runs for each experiment and we report averaged results.

As a benchmark we used the following text classification task: predict all the topic codes associated with an article using the first τ tokens in the article. We used the *Reuters-21578*[4] dataset. This corpus contains 10788 news documents classified into 90 topics. We used the provided training/test split. For each document, we considered the first $\tau = 250$ tokens as input text for the CNN. Note that, in contrast to other previous work (Li et al., 2016), we consider all topics and formulate a multi-label classification problem. For each test article we computed precision, recall and F1 score comparing the actual topic codes and those predicted by the CNN. As evaluation metrics we used the average F1 score and the average preci-

[2] https://emorynlp.github.io/nlp4j

[3] Source code available at https://github.com/thomsonreuters/attr2vec

[4] http://www.nltk.org/book/ch02.html

sion across all test articles.

We trained multiple CNN models, using either word vectors ($\vec{w}$) or the concatenation of word and POS tag vectors ($\vec{w}\frown\vec{p}$) as inputs, and keeping these vectors *static* throughout training or allowing the CNN to update them via backpropagation (*non-static*). We also considered logistic regression as baseline method, using averaged vectors calculated over the input text as features, as in Zhang and Wallace (2015). As this is a multilabel task, we used the one-vs-all formulation of logistic regression, which attempts to fit one classifier per class with each class being fitted against all other classes. L_1 regularization was applied with a weight of 0.005.

Results Table 1 reports the result of our experiments. Each entry shows the average F1 score and the average precision in parentheses.

First note that the CNN model consistently outperforms logistic regression for all considered settings. The CNN performance improves if it receives as input pre-trained vectors as opposed to random ones, consistently with other works (Kim, 2014). The performance is comparable when GloVe or attr2vec word vectors are used as input.

The key advantage of our attr2vec model over GloVe is demonstrated when additional contextual information is considered in the CNN model. The performance of the CNN model improves if POS vectors are considered together with GloVe word vectors in input, both when such POS vectors are randomly initialized ($\vec{w_i}\frown\vec{p_r}$) and independently trained with the GloVe model ($\vec{w_i}\frown\vec{p_i}$). However, the best performance is achieved when word and POS tags vectors are jointly trained with our attr2vec model ($\vec{w_j}\frown\vec{p_j}$).

Note that the aim of the paper was not to show that POS tags help for text classification tasks (to that end, an exhaustive exploration of the parameter space would have been needed); instead, the goal of this work is to introduce a new embedding model that jointly learns a representation for words and POS tags, capturing the interaction between them, and to show that such representation is beneficial for a CNN with respect to embeddings learned in an independent fashion, given the same network settings. Standard embedding models (like GloVe), in fact, can capture either the interactions between words or between POS tags in an independent fashion. Our attr2vec model, in addition, captures the cross-interaction between

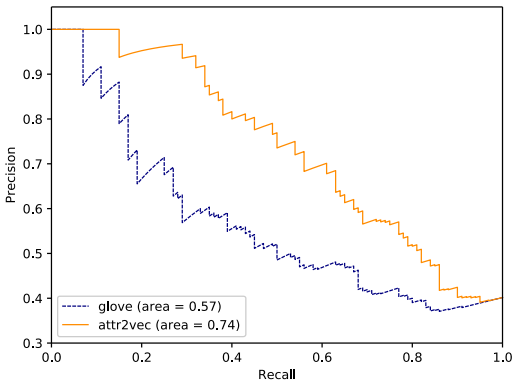

Figure 3: Recall-precision curve when attempting to rank the similar words above the related ones on the *WordSim353* dataset.

words and contextual attributes, jointly learning their representation, and our results suggest that this additional information is beneficial for the performance of the CNN model. Moreover, note that our attr2vec algorithm, unlike GloVe, can handle generic contextual information.

4.3 Word Similarity Experiment

In our second experiment we wanted to address if our attr2vec model was able to produce dependency-based embeddings that exhibit more functional similarity than GloVe embeddings (that usually yield broad topical similarities). To this end, we trained 200-dimensional attr2vec vectors using a dependency-based approach - i.e, each row in the feature matrix consist of a word and a dependency label (see the example in Figure 2).

Our evaluation closely follows the one in Levy and Goldberg (2014). In particular, we used the *WordSim353* dataset (Finkelstein et al., 2001; Agirre et al., 2009) containing pairs of similar words that reflect either *relatedness* (topical similarity) or *similarity* (functional similarity) relations. The pairs are ranked according to the cosine similarity between the corresponding word vectors. The idea is that a model that focuses on functional similarity should rank similar pairs in the dataset above the related ones. For instance, such a model should rank the pair *money-currency* (i.e., functionally similar words) above the pair *money-laundering* (i.e., topically similar words). We drew a recall-precision curve by considering related pair as a miss and similar pair as a hit. In this way we aimed to capture the embeddings affinity towards the *similarity* subset over the *re-*

latedness one.

Figure 3 reports the result of the experiment. The attr2vec curve (orange solid line) is higher than the GloVe one (blue dashed line) and the area under the cuve is larger (0.74 with respect to 0.57), suggesting that attr2vec yields more functional similarities with respect to GloVe. Note that a similar behaviour has been observed in Levy and Goldberg (2014) for context-predictive models (i.e., the skip-gram model with negative sampling). To the best of our knowledge, attr2vec is the first model that incorporates syntactic dependency relations in a co-occurrence counts based model (such as GloVe). Moreover, attr2vec is a general model that can handle additional arbitrary contextual information.

4.4 Qualitative Evaluation

Our final evaluation is qualitative. We trained 200-dimensional attr2vec embeddings using news topics as additional contextual information and a linear bag-of-words approach - i.e., each row in the feature matrix consists of a pair of words and the topic of the news article where such pair has been observed (see the first and the last group of columns for the example in Figure 1). In particular, we used the same collection of $\sim 8M$ news articles presented in Section 4.1 and we considered the following two article topics: general news stories (G) and sport news (SPO).

Figure 4 shows a two-dimensional projection of the 200-dimensional vector space where words and topics representations lie, obtained using the *t-SNE*[5] visualisation technique (Maaten and Hinton, 2008). Here the two topic points (G on the left and SPO on the right of the figure) seem to metaphorically act as "magnets", modifying the space and forming two clusters of words. The left cluster around the representation of topic G includes general words not related to sports such as "mars", "sound", "warranty", "finance", "train", while the right cluster around the representation of topic SPO contains words related to sports such as "football", "coach", "game", "stadium", "cricket". Words that are related with both general news stories and sport news lie somewhere in the middle between these two clusters. Examples of such words include "penalties", "transfer", "medical", "goal", "supporters". Note that there are other attractive and repulsive forces in the vector space driven by word similarity, and that a two-dimensional representation is only able to capture a small portion of all relations that take place in the higher dimensional space.

[5]We used the TensorBoard implementation of *t-SNE*.

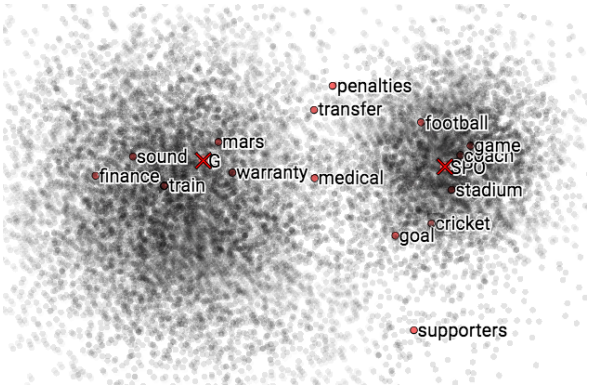

Figure 4: Two-dimensional projection of the 200-dimensional vector space, that contains representations of both words and topics, using *t-SNE*. In particular, we considered two topics: general news stories (G) and sport news (SPO).

5 Conclusions

In this paper, we proposed attr2vec, a novel embedding model that can jointly learn a distributed representation for words and contextual attributes. Our model is general and can handle multiple arbitrary contextual information simultaneously. To do so, we defined a novel loss function based on factorization machines. Moreover, attr2vec can mimic existing word embedding algorithms when no additional contextual information is considered. In particular, GloVe is a special case of our model.

We have presented an experimental study where we considered POS tags as additional contextual information, and fed a convolutional neural network (CNN) with both word and POS tag vectors. The results suggest that the CNN prediction performance improves when word and context vectors are jointly learned by our attr2vec model. In addition, we described how to train dependency-based attr2vec embeddings and showed that they produce different kinds of similarities. We also provided some insights into how the vector space is affected by contextual attributes, which seem to act like "magnets" that attract or repulse words, that are themselves subject to attractive or repulsive forces driven by similarity.

While attr2vec benefits from structural information, it has a price: the number of features is in-

creased, and the computational cost is increased compared to a model that does not use contextual information. Each additional attribute may furthermore introduce its own noise (component-specific errors) into the process. Nevertheless, the overall improvement can help in tasks where quality is of the utmost importance and high-quality annotation components are available.

In future work, we aim to investigate the effect of adding different contextual information, and we plan to test the resulting models in various applications.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological Priors for Probabilistic Neural Word Embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Denny Britz. 2015. Implementing a CNN for Text Classification in Tensorflow. https://github.com/dennybritz/cnn-text-classification-tf.

Danqi Chen and Christopher D. Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Jinho D Choi. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jinho D Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Luciano del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological Word-Embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2017. Neural Machine Translation by Generating Multiple Linguistic Factors. In *International Conference on Statistical Language and Speech Processing*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and SVN Vishwanathan. 2016. Wordrank: Learning word embeddings via robust ranking. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* .

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the eighteenth conference on computational natural language learning*.

Jiwei Li and Dan Jurafsky. 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding? *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* .

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative Topic Embedding: a Continuous Representation of Documents. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9(Nov):2579–2605.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The Role of Context Types and Dimensionality in Learning Word Embeddings. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*. pages 2265–2273.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Fabio Petroni, Luciano Del Corro, and Rainer Gemulla. 2015. CORE: Context-Aware Open Relation Extraction with Factorization Machines. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-Conflated Semantic Representations. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* .

Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3(3):57:1–57:22.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast Context-aware Recommendations with Factorization Machines. In *Proceedings of the 34th international ACM SIGIR Conference on Research and development in Information Retrieval*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Mikhail Trofimov and Alexander Novikov. 2016. tffm: TensorFlow implementation of an arbitrary order Factorization Machine. https://github.com/geffy/tffm.

Albert Weichselbraun, Stefan Gindl, and Arno Scharl. 2013. Extracting and Grounding Context-Aware Sentiment Lexicons. *IEEE Intelligent Systems* 28(2):39–46.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Ye Zhang, Md Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, and Matthew Lease. 2017. Neural Information Retrieval: A Literature Review. *arXiv preprint arXiv:1611.06792v3* .

Ye Zhang and Byron Wallace. 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *arXiv preprint arXiv:1510.03820* .