2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Neural Word Representations from Large-Scale Commonsense Knowledge

Jiaqiang Chen
IIIS, Tsinghua University
Beijing, China
Email: cjq0707@gmail.com

Niket Tandon
Max Planck Institute for Informatics
Saarbrücken, Germany
Email: ntandon@mpi-inf.mpg.de

Gerard de Melo
IIIS, Tsinghua University
Beijing, China
Email: gdm@demelo.org

***Abstract*—There has recently been a surge of research on neural network-inspired algorithms to produce numerical vector representations of words, based on contextual information. In this paper, we present an approach to improve such word embeddings by first mining cognitively salient word relationships from text and then using stochastic gradient descent to jointly optimize the embeddings to reflect this information, in addition to the regular contextual information captured by the word2vec CBOW objective. Our findings show that this new training regime leads to vectors that better reflect commonsense information about words.**

I. Introduction

Words are substantially discrete in nature, and hence, traditionally, the vast majority of natural language processing tools, including statistical ones, have regarded words as distinct atomic symbols. In recent years, however, the idea of embedding words in a vector space using neural network-inspired algorithms has gained enormous popularity. Mapping words to vectors in a way that reflects word similarities provides machine learning algorithms with much-needed generalization ability. If the words *car* and *automobile* have similar vectors, a learning algorithm is better equipped to generalize from one word to the other. Word embeddings are typically trained using large amounts of contextual data. While regular sentence contexts play a vital role in meaning acquisition, words and concepts are often also acquired by other means. Humans may pay special attention to certain cognitively salient features of an object, or rely on more explicit definitions (e.g., looking up a meaning online).

In this paper, we propose a model to jointly train word representations not just on regular contexts, as in the word2vec CBOW model [1], but also to reflect more salient information. For the latter, we use *information extraction* techniques [2] on large-scale text data to mine definitions and synonyms as well as lists and enumerations. Rather than considering all contexts as equal, our approach can be viewed as treating certain specific contexts as more informative than others. Consider the sentence *The Roman Empire was remarkably multicultural, with "a rather astonishing cohesive capacity" to create a sense of shared identity....* While it contains several useful signals, *Roman* does not seem to bear an overly close relationship with *capacity*, *astonishing*, or *sense*. In contrast, upon encountering *Greek and Roman mythology*, we may conclude that *Roman* and *Greek* are likely related. Our training objective thus encourages saliently related words to have similar representations.

This research was supported by China 973 Program Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 61450110088.

II. Related Work

Many of the current methods for obtaining distributed word embeddings are neural network-inspired and aim at rather dense real-valued vector spaces. While early work focused on probabilistic language models [3], Collobert et al. [4] used a convolutional neural network to maximize the difference between scores from text windows in a large training corpus and corresponding randomly generated negative examples. Mikolov et al. [1] proposed simplified network architectures to efficiently train such vectors at a much faster rate and thus also at a much larger scale. Their word2vec[1] implementation provides two architectures, the CBOW and the Skip-gram models. CBOW also relies on a window approach, attempting to use the surrounding words to predict the current target word. However, it simplifies the hidden layer to be just the average of surrounding words' embeddings. The Skip-gram model tries to do the opposite. It uses the current word to predict the surrounding words. In our approach, we build on the CBOW variant, as its optimization runs faster.

There have been other proposals to adapt the word2vec models. Levy et al. [5] use dependency parse relations to create word embeddings that are able to capture contextual relationships between words that are further apart in the sentence. Further analysis revealed that their word embeddings capture more functional but less topical similarity. Faruqui et al. [6] apply post-processing steps to existing word embeddings in order to bring them more in accordance with semantic lexicons. Rather than using rich structured knowledge sources, our work focuses on improving word embeddings using textual data, by relying on information extraction to expose particularly valuable contexts and relationships in a text corpus. In particular, we are not aware of any previous work that mines large-scale common-sense knowledge to train embeddings of lexical units.

III. Salient Proximity Model

Our approach is to simultaneously train the word embeddings on generic contexts from the corpus on the one hand and on semantically significant contexts, obtained using extraction techniques, on the other hand. For the regular general contexts, we draw on the word2vec CBOW model [1] to predict a word given its surrounding neighbors in the corpus.

[1] https://code.google.com/p/word2vec/

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.150

At the same time, our model relies on our ability to extract semantically salient contexts that are more indicative of word meanings. These extractions will be described in detail later in Section IV. Our algorithm assumes that they have been transformed into a set of word pairs likely to be closely related, which are used to modify the word embeddings. Due to this more focused information, we expect the final word embeddings to reflect more semantic information than embeddings trained only on regular contexts. Given an extracted pair of related words, the intuition is that the embeddings for the two words should be pulled together. Given a word w_{t}, our objective function attempts to maximize the probability of finding its related words w_{r}:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w_{\mathrm{r}}}\log p(w_{\mathrm{r}}|w_{\mathrm{t}}) \tag{1}$$

Here, T is the vocabulary size and the probabilities are modeled using a softmax, defined as follows:

$$p(w_{\mathrm{r}}|w_{\mathrm{t}}) = \frac{\exp(V_{w_{\mathrm{r}}}^T \cdot V_{w_{\mathrm{r}}})}{\sum_{w_{\mathrm{r}'}}\exp(V_{w_{\mathrm{r}'}}^T \cdot V_{w_{\mathrm{t}}})} \tag{2}$$

$V_{w_{\mathrm{r}}}$, $V_{w_{\mathrm{t}}}$ refer to the word vectors for two related words w_{r}, w_{t}, while $w_{\mathrm{r}'}$ with corresponding vectors $V_{w_{\mathrm{r}'}}$ refer to all possible words. We use the inner product to score how well two words match. When they are very similar or related, their embeddings should be close to each other and hence the inner product of their embeddings should be large. Using the softmax function, we can take a maximum likelihood approach to train the embeddings in a tractable manner. However, the gradient is as follows:

$$\begin{aligned}\frac{\partial \log p(w_{\mathrm{r}}|w_{\mathrm{t}})}{\partial V_{w_{\mathrm{t}}}} &= \frac{\partial V_{w_{\mathrm{r}}}^T \cdot V_{w_{\mathrm{t}}}}{\partial V_{w_{\mathrm{t}}}} - \frac{\log\sum_{w_{\mathrm{r}'}}\exp(V_{w_{\mathrm{r}'}}^T \cdot V_{w_{\mathrm{t}}})}{\partial V_{w_{\mathrm{t}}}} \\ &= V_{w_{\mathrm{r}}} - \sum_{w_{\mathrm{r}'}}\frac{\exp(V_{w_{\mathrm{r}'}} \cdot V_{w_{\mathrm{r}}})}{\sum_{w_{\mathrm{r}'}}\exp(V_{w_{\mathrm{r}'}}^T \cdot V_{w_{\mathrm{t}}})}V_{w_{\mathrm{r}'}} \\ &= V_{w_{\mathrm{r}}} - \sum_{w_{\mathrm{r}'}} p(w_{\mathrm{r}'}|w_{\mathrm{t}})V_{w_{\mathrm{r}'}} \\ &= V_{w_{\mathrm{r}}} - \mathbb{E}_p V_{w_{\mathrm{r}'}}\end{aligned} \tag{3}$$

To compute this gradient, the expectation of all the word vectors with respect to their probabilities would be needed. The time complexity for this is proportional to the vocabulary size, which is typically very large. Here, we use negative sampling as a speed-up technique [7]. This can be viewed as a simplified version of Noise Contrastive Estimation (NCE) [8], which reduces the problem of determining the softmax to that of binary classification, discriminating between samples from the data distribution and negative samples. In particular, we consider a distribution of random noise and optimize for discriminating between the positive examples and the noise. This can be further simplified by removing certain weights before each term in the objective function. With negative sampling, the objective for each training pair is then as follows:

$$\log(\sigma(V_{w_{\mathrm{r}}}^T \cdot V_{w_{\mathrm{t}}})) + \sum_{i=1}^{k}\mathrm{E}_{w_{\mathrm{i}}\sim P_n(w)}\log(\sigma(-V_{w_{\mathrm{i}}}^T \cdot V_{w_{\mathrm{t}}})) \tag{4}$$

Here, $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, and $V_{w_{\mathrm{t}}}$, $V_{w_{\mathrm{r}}}$ refer to the vectors for the two words w_{t} and w_{r}, while w_{i} refers to randomly chosen words.

Thus, we generate negative samples w_{i} from some known noise distribution $P_n(\mathrm{w})$ and treat them, together with w_{t}, as negative sample pairs. We attempt to maximize the score for the positive training data and minimize the score of the negative samples. In the training procedure, this amounts to simply generating k random negative samples for each extracted word pair. That is, we replace w_{r} with random words from the vocabulary. For the negative samples, we assign the label $l = 0$, whereas for the original word pairs, $l = 1$. Now, for each word pair we try to minimize its loss L:

$$L = -l\,\log f - (1-l)\,\log(1-f) \tag{5}$$

$$f = \sigma(v_{w_{\mathrm{t}}}^T \cdot v_{w_{\mathrm{r}}}) \tag{6}$$

We use stochastic gradient descent to optimize this function. The formulae for the gradient are easy to compute:

$$\begin{aligned}\frac{\partial L}{\partial V_{w_{\mathrm{r}}}} &= -l\,\frac{1}{f}\,f\,(1-f)\,V_{w_{\mathrm{t}}} + (1-l)\,\frac{1}{1-f}\,f\,(1-f)\,V_{w_{\mathrm{t}}} \\ &= -(l-f)\,V_{w_{\mathrm{t}}}\end{aligned} \tag{7}$$

$$\frac{\partial L}{\partial V_{w_{\mathrm{t}}}} = -(l-f)\,V_{w_{\mathrm{r}}} \tag{8}$$

This objective is optimized alongside with the original word2vec CBOW objective, i.e., our overall model combines the two objectives. Implementation-wise, we train the model in parallel with the CBOW model, which allows us to inject the extracted knowledge into the word vectors such that they are reflected during the CBOW training rather than just as a post-processing step. Thus we obtain a joint learning process in which the two components are able to mutually influence each other. Both objectives contribute to the embeddings' ability to capture semantic relationships. Training with the extracted contexts enables us to adjust word embeddings based on concrete evidence of semantic relationships, while the use of general corpus contexts enables us to maintain the advantages of the word2vec CBOW model, in particular its ability to benefit from massive volumes of raw corpus data.

IV. Information Extraction

Our model can flexibly incorporate semantic relationships extracted using various kinds of information extraction methods. Different kinds of sources and extraction methods can bring different sorts of information to the vectors, suitable for different applications. In our experiments, we investigate two main sources: a dictionary corpus from which we extract definitions and synonyms, and a general Web corpus, from which we extract lists. Our model could similarly be used with other extraction methods.

A. Definition and Synonym Extraction

Surely, any sizeable, broad-coverage Web corpus will contain significant occurrences of word definitions, e.g. whenever new terminology is introduced. These can be harvested using broad-coverage Definition Extraction methods [9]. Instead of adopting such generic methods intended to operate on arbitrary text, another option, appropriate for Web corpora, is to specifically identify the kinds of Web sources that provide high-quality definitions (e.g. Wiktionary or Wikipedia). In fact, when compiling a Web corpus with the specific purpose of using it to train word representations, one may reasonably wish to explicitly ensure that it includes dictionaries available on

the Web. In our experiments, we use the GNU Collaborative International Dictionary of English (GCIDE) as our dictionary corpus, which is derived from an out-of-copyright edition of Webster's Revised Unabridged Dictionary. From this data, we extract the dictionary glosses as definitions, as well as synonyms. We ignore embedded tags within the definitions and synonym entries, as these are used to mark usage notes and other attributes. In total, we obtain 208,881 definition entries. Some words have multiple meanings and thus are part of several entries. We also obtain 10,148 synonym entries, each of which consists of one or more synonyms for a given word.

B. List Extraction

Lists and enumerations are another promising source of information. Words co-occurring together within a list are not just semantically connected but often even of the same type. These sorts of contexts thus also have the potential to improve the word embeddings. We extract them from the UKWaC corpus [10], as it is pre-annotated with POS tags, which simplifies our extraction process. To extract lists of similar words, we use a simple rule-based method. We first search for continuous appearances of commas, which indicate a possible list of similar items. To filter out noise, we require that the entries in the list be approximately of equal length and that the length of each entry be in the range from 1 to 4 words, as longer entries are much more likely to be short sentences or clauses. We also restrict list items to be nouns or adjectives using the given POS tags. Additionally, we rely on a few special search patterns such as: "$\langle x \rangle$ and $\langle y \rangle$", "$\langle x \rangle$ or $\langle y \rangle$", "include $\langle x_1 \rangle$, ..., (and) $\langle x_n \rangle$", "$\langle$noun$\rangle$ like $\langle x_1 \rangle$, ..., (and) $\langle x_n \rangle$", "$\langle$noun$\rangle$ such as $\langle x_1 \rangle$, ..., (and) $\langle x_n \rangle$". In total, 339,111 lists are extracted from the UKWaC, examples of which are shown in Table I. Despite some noise, the words in the lists tend to be of the same or similar type and represent similar or related concepts.

TABLE I. LISTS OF RELATED WORDS EXTRACTED FROM UKWAC

player, captain, manager, director, vice-chairman
group, race, culture, religion, organisation, person
prehistoric, roman, early-medieval, late-medieval, post-medieval, modern
ballscrews, leadscrews, worm gear, screwjacks, linear, actuator
Cleveland, Essex, Lincolnshire, Northamptonshire, Nottinghamshire, Thames Valley, South Wales
David, Roberto, Pano, Ian, Andy, Kevin, Alistair, Tomas, Stefan, Gary, Paul, Gary
banking, mortgage, loan, credit card, insurance, pension
French, German, Spanish, traditional Chinese, simplified Chinese, Dutch, Swedish, Italian, English, Korean, Portuguese, Japanese, Arabic
Buddhism, Christianity, Hinduism, Islam, Judaism
ant.py, dimdriver.py, dimdriverdatafile.py, dimdriverdatasetdef.py, dimexception.py, dimmaker.py, dimoperators.py, dimparser.py, dimrex.py, dimension.py
athletics, badminton, cycling, swimming, judo, gymnastics, tennis, football, handball, volleyball

V. EXPERIMENTS

Data. For semantically salient contexts, we rely on the data and extraction techniques described above in Sections IV-A and IV-B to obtain pairs of related words from definitions, synonyms, and lists. For the regular contexts used by the CBOW model, we rely on a frequently used 2010 Wikipedia data set[2]. We normalize the text to lower case and remove special characters, obtaining 1,205,009,210 tokens after this preprocessing. We select all words appearing at least 50 times, yielding a vocabulary size of 220,521.

Training. The extracted contexts are used to train word embeddings jointly with the original word2vec model. Our implementation relies on a multi-threaded architecture in which some threads optimize for the word2vec objective, training on different parts of the corpus. At the same time, alongside with these threads, further threads optimize based on the extracted pairs of words using the objective given earlier. All threads asynchronously update the word embeddings, using stochastic gradient descent steps. Thus, the different kinds of components can mutually influence each other. We use 20 threads for the CBOW architecture, and set the window size of the CBOW model to 8. We run it for 3 passes over the Wikipedia data set, which is sufficient to achieve good results. We sample 10 random words as negative examples for each instance. Additional threads are used for the extracted pairs of words. We use 4 threads each for lists and definitions (by splitting definitions) and one thread for synonyms. In each case, the extractions lead to positive pairs of semantically related words. For definitions and synonyms, the word pair consists of a headword and one word from its definition, or of the headword and one of its synonyms. For the list extraction setup, the training word pairs consist of any two words from the same list. For these word pairs, we also randomly sample 10 words as the corresponding negative examples.

Different learning rates allow us to balance each source's contribution to the word embeddings. We set the initial learning rate for the CBOW threads to be 0.05 and report results for different rates for the other threads, ranging from 0.001 to 0.1. To ensure convergence, we stop the training when all the CBOW threads finish – The other threads are terminated only then. This ensures that we are always training on both components of the model jointly rather than allowing one component to dominate towards the end. This also makes sense because the extractions are only supplementary to the CBOW model.

Evaluation and Analysis. We use the standard wordsim-353 [11] dataset, as well as the significantly larger MEN [12] dataset to assess the semantic similarities reflected in the final word embeddings. These contain English word pairs with similarity judgements elicited from human assessors. We calculate the cosine distance of word embeddings for the word pairs in these datasets and compare them to the scores from the human annotations using Spearman's ρ.

First, we test the effect of the word embeddings' dimensionality on the performance of the model, using the definitions and synonyms. The dimension of the word vectors ranges from 50 to 500. We fix the learning rate to be 0.01 and test on the wordsim-353 data set. The results are plotted in Fig. 1. We observe that as the vector dimensionality increases, we achieve better correlations with the human assessors, with a slight drop-off towards the end. In the following experiments, we set the dimensionality to 200, which yields good results while keeping the training time reasonable.

Fig. 2 plots results on the wordsim-353 dataset when varying the learning rate α for the additional threads. Even for a rate as low as 0.001, we can obtain some improvement over the CBOW baseline (which corresponds to an α setting of 0.0).

[2]http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2

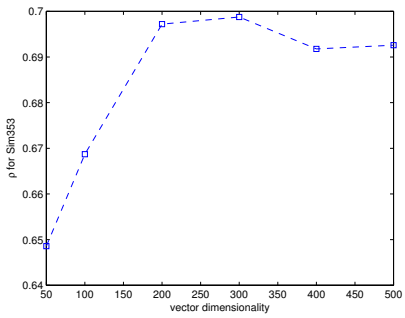

Fig. 1. Effect of dimensionality on Spearman's ρ of the wordsim-353 dataset

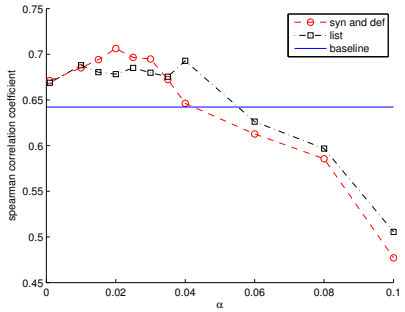

Fig. 2. Spearman's ρ on the wordsim-353 dataset

As α increases, the result gets better. The best result we get for synonyms and definitions is 0.706 (for $\alpha = 0.02$), whereas for lists from UKWaC, it is 0.693 (for $\alpha = 0.04$). Both lead to noticeably better results than the CBOW baseline's ρ score of 0.642. For large α, the augmentation performs worse than the baseline. This is expected, as an overly high α causes information from the extractions to overwhelm the original CBOW model, leading to excessively biased final embeddings.

Fig. 3 plots the results on the MEN dataset. The best-performing learning rate is different from that for wordsim-353. In particular, well-performing learning rates are slightly smaller. For the definitions and synonyms, the best is 0.002, while for the lists, the best learning rate is 0.020. Again, joint training outperforms the baseline.

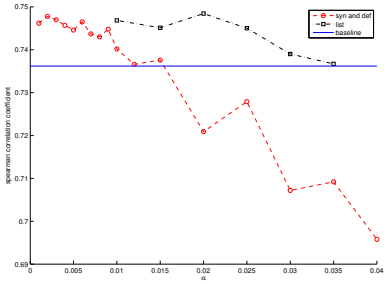

Fig. 3. Spearman's ρ on the MEN dataset

TABLE II. BEST RESULTS FOR THE WORD SIMILARITY TASKS

	wordsim-353	MEN
CBOW (baseline)	0.642	0.736
Definitions and synonyms	0.706	0.748
Lists	0.693	0.749

Table II summarizes the best results that we obtain on the two similarity tasks.[3] We obtain higher correlation coefficients than the baseline, suggesting that the word vectors capture more semantic properties of words and thus should be of value in applications that benefit from semantic information.

VI. CONCLUSION

In this paper, we have proposed a novel method to obtain better distributed representations of concepts, drawing on the idea that certain contexts exhibit more cognitively salient information than others. We rely on large-scale information extraction to mine such information, focusing on word definitions and synonyms as well as lists and enumerations. Our model extends the well-known CBOW model to capture not just raw text but also salient semantic proximity information. Overall, our results suggest that information extraction (and knowledge-based methods [13]) can lead to improved word vectors, making them useful for semantic applications and calling for further research in this area.

REFERENCES

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[2] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of COLING*, 1992, pp. 539–545.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *JMLR*, vol. 3, pp. 1137–1155, 2003.

[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *JMLR*, vol. 12, pp. 2493–2537, 2011.

[5] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of ACL*, 2014, pp. 302–308.

[6] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," in *Proceedings of NAACL*, 2015.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, 2013.

[8] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of ICML*, 2012.

[9] G. Sierra, M. Pozzi, and J.-M. Torres, Eds., *Proceedings of the 1st Workshop on Definition Extraction*, 2009.

[10] M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta, "The WaCky wide web: a collection of very large linguistically processed web-crawled corpora," *LangResources & Eval.*, vol. 43, no. 3, pp. 209–226, 2009.

[11] L. Finkelstein, G. Evgenly, M. Yossi, R. Ehud, S. Zach, W. Gadi, and R. Eytan, "Placing search in context: the concept revisited," in *Proceedings of WWW*, 2001.

[12] E. Bruni, N. K. Tran, and M. Baroni, "Multimodal distributional semantics," *J. Artif. Int. Res.*, vol. 49, no. 1, pp. 1–47, 2014.

[13] G. de Melo, "Wiktionary-based word embeddings," in *Proceedings of MT Summit XV*, 2015.

[3] Unfortunately, there is no independent tuning set from the same distribution and thus we follow previous work in reporting best results on the final set.