Explaining and Generalizing Skip-Gram through Exponential Family Principal Component Analysis

Ryan Cotterell  Adam Poliak  Benjamin Van Durme  Jason Eisner
Center for Language and Speech Processing
Johns Hopkins University
{ryan.cotterell,azpoliak,vandurme,jason}@cs.jhu.edu

Abstract
The popular skip-gram model induces word embeddings by exploiting the signal from word-context cooccurrence. We offer a new interpretation of skip-gram based on exponential family PCA—a form of matrix factorization. This makes it clear that we can extend the skip-gram method to tensor factorization, in order to train embeddings through richer higher-order cooccurrences, e.g., triples that include positional information (to incorporate syntax) or morphological information (to share parameters across related words). We experiment on 40 languages and show that our model improves upon skip-gram.

1 Introduction
Over the past years NLP has witnessed a veritable frenzy on the topic of word embeddings: low-dimensional representations of distributional information. The embeddings, trained on extremely large text corpora such as Wikipedia and Common Crawl, are claimed to encode semantic knowledge extracted from large text corpora.

Numerous methods have been proposed—the most popular being skip-gram (Mikolov et al., 2013) and GloVe (Pennington et al., 2014)—for learning these low-dimensional embeddings from a bag of contexts associated with each word type. Natural language text, however, contains richer structure than simple context-word pairs. In this work, we embed n-tuples rather than pairs, allowing us to escape the bag-of-words assumption and encode richer linguistic structures.

As a first step, we offer a novel interpretation of the skip-gram model (Mikolov et al., 2013). We show how skip-gram can be viewed as an application of exponential-family principal components analysis (EPCA) (Collins et al., 2001) to an integer matrix of cooccurrence counts. Previous work has related the negative sampling estimator for skip-gram model parameters to the factorization of a matrix of (shifted) positive pointwise mutual information (Levy and Goldberg, 2014b). We show the skip-gram objective is just EPCA factorization.

By extending EPCA factorization from matrices to tensors, we can consider higher-order cooccurrence statistics. Here we explore incorporating positional and morphological content in the model by factorizing a positional tensor and morphology tensor. The positional tensor directly incorporates word order into the model, while the morphology tensor adds word-internal information. We validate our models experimentally on 40 languages and show large gains under standard metrics.¹

2 Matrix Factorization
In this section, we briefly explain how skip-gram is an example of EPCA. We are given data in the form of a matrix $X \in \mathbb{R}^{n_1 \times n_2}$, where $X_{ij}$ is the number of times that word $j$ appears in context $i$ under some user-specified definition of “context.”

Principal components analysis (Pearson, 1901) approximates $X$ as the product $C^T W$ of two matrices $C \in \mathbb{R}^{d \times n_1}$ and $W \in \mathbb{R}^{d \times n_2}$, whose columns are $d$-dimensional vectors that embed the contexts and the words, respectively, for some user-specified $d < \min(n_1, n_2)$. Specifically, PCA minimizes $^2$

$$\|X - C^T W\|_F^2 = \sum_{ij} (X_{ij} - c_i \cdot w_j)^2 \quad \text{ (1)}$$

$$= \sum_j \|x_j - C^T w_j\|^2 \quad \text{ (2)}$$

where $c_i$, $w_j$, $x_j$ denote the $i^{th}$ column of $C$ and the $j^{th}$ columns of $W$ and $X$, and $c_i \cdot w_j$ denotes an inner product of vectors (sometimes called “cosine

¹The code developed is available at https://github.com/azpoliak/skip-gram-tensor.
²Singh and Gordon (2008) offer a comprehensive discussion of PCA and other matrix factorization techniques in ML.
EPCA allows us to suppose that each \( \mathbf{x}_j \) was drawn from a multinomial—a more appropriate family for drawing a count vector. Our observation is that skip-gram is precisely \textbf{multinomial EPCA with the canonical link function} (Mohamed, 2011), where the count vector \( \mathbf{x}_j \) comes from a multinomial with canonical parameters \( C^\top \mathbf{w}_j \). It chooses embeddings \( C, W \) to maximize
\[
\sum_j \sum_i X_{ij} \log p(\text{context } i \mid \text{word } j)
\]
\[
= \sum_j \sum_i X_{ij} \log \frac{\exp(c_i \cdot w_j)}{\sum_{i'} \exp(c_{i'} \cdot w_j)}
\]
This is the log-likelihood (plus a constant) if we assume that for each word \( j \), the context vector \( \mathbf{x}_j \) was drawn from a multinomial with canonical parameter vector \( C^\top \mathbf{w}_j \) and count parameter \( N_j = \sum_i X_{ij} \). This is the same model as in Figure 1a, but with a different conditional distribution for \( \mathbf{x}_j \), and with \( \mathbf{x}_j \) taking an additional observed parent \( N_j \) (which is the token count of word \( j \)).

### 2.1 Related work

Levy and Goldberg (2014b) also interpreted skip-gram as matrix factorization. They argued that skip-gram estimation by negative sampling implicitly factorizes a shifted matrix of positive empirical pointwise mutual information values. We instead regard the skip-gram objective itself as demanding EPCA-style factorization of the count matrix \( X \): i.e., \( X \) arose stochastically from some unknown matrix of multinomial parameters (column \( j \) of \( X \) generated from parameter column \( j \) ), and we seek a rank-\( d \) estimate \( C^\top W \) of that matrix.

pLSI (Hofmann, 1999) similarly factors an unknown matrix of multinomial probabilities, which is \textbf{multinomial EPCA with the identity link function}. In contrast, our unknown matrix holds the “canonical parameters” for a multinomial—arbitrarily shifted log-probabilities.

Our EPCA interpretation applies equally well to the component distributions that are used in hierarchical softmax (Morin and Bengio, 2005), which is an alternative to negative sampling. Additionally, it yields avenues of future research using Bayesian (Mohamed et al., 2008) and maximum-margin (Srebro et al., 2004) extensions to EPCA.

The canonical parameters of the distribution. We use the so-called canonical link, meaning that these two steps are inverses of each other and thus the canonical parameters are themselves a linear function of \( w_j \).

It can also be seen as a conditional log-linear model with conjunctive feature weights \( C \) and learned atomic features \( W \).
3 Tensor Factorization

Having seen that skip-gram is a form of matrix factorization, we can generalize it to tensors. In contrast to the matrix case, there are several distinct definitions of tensor factorization (Kolda and Bader, 2009). We focus on the polyadic decomposition (Hitchcock, 1927), which yields a satisfying generalization. The tensor analogue to PCA is rank-d tensor approximation, which minimizes

\[
\|X - C \otimes W \otimes R\|_F^2
\]

\[
\sum_{ijk} (X_{ijk} - 1 \cdot (c_i \otimes w_j \otimes r_k))^2
\]  

(5)

\[
\sum_{j} \left\| X_{-jk} - C^\top (w_j \otimes r_k) \right\|^2
\]  

(6)

Given a tensor \(X \in \mathbb{R}^{n_1 \times n_2 \times n_3}\), this objective tries to predict each entry as the three-way dot product of the columns \(c_i, w_j, r_k \in \mathbb{R}^d\), thus finding an approximation to \(X\) that factorizes into \(C, W, R\). This polyadic decomposition of the approximating tensor can be viewed as a Tucker decomposition (Tucker, 1966) that enforces a diagonal core.

In our setting, the new matrix \(R \in \mathbb{R}^{d \times n_3}\) embeds types of context-word relations. The tensor \(X\) can be regarded as a collection of \(n_2n_3\) count vectors \(X_{jk} \in \mathbb{N}^{n_1}\): the fibers of the tensor, each of which provides the context counts for some (word \(j\), relation \(k\)) pair. Typically, \(X_{jk}\) counts which context words \(i\) are related to word \(j\) by relation \(k\).

We now move from third-order PCA to third-order EPCA. Minimizing equation (6) corresponds to maximum-likelihood estimation of the graphical model in Figure 1b, in which each fiber of \(X\) is viewed as being generated from a Gaussian all at once. Our higher-order skip-gram (HOSG) replaces this Gaussian with a multinomial. Thus, HOSG attempts to maximize the log-likelihood

\[
\sum_{ijk} X_{ijk} \log p(\text{context } i \mid \text{word } j, \text{relation } k)
\]

\[
= \sum_{ijk} X_{ijk} \log \frac{\exp \left( 1 \cdot (c_i \otimes w_j \otimes r_k) \right)}{\sum_{i'} \exp \left( 1 \cdot (c_{i'} \otimes w_j \otimes r_k) \right)}
\]  

(7)

Note that as before, we are taking the total count \(N_{jk} = \sum_i X_{ijk}\) to be observed. So while our embedding matrices must predict which words are related to word \(j\) by relation \(k\), we are not probabilistically modeling how often word \(j\) participates in relation \(k\) in the first place (nor how often word \(j\) occurs overall). A simple and natural move in future would be to extend the generative model to predict these facts also from \(w_j\) and \(r_k\), although this weakens the pedagogical connection to EPCA.

We locally optimize the parameters of our probability model—the word, context and relation embeddings—through stochastic gradient ascent on (7). Each stochastic gradient step computes the gradient of a single summand \(X_{ijk} \log p(i \mid j, k)\). Unfortunately, this requires summing over \(n_1\) contexts in the denominator of (8), which is problematic as \(n_1\) is often very large, e.g., \(10^7\). Mikolov et al. (2013) offer two speedup schemes: negative sampling and hierarchical softmax. Here we apply the negative sampling approximation to HOSG; hierarchical softmax is also applicable. See Goldberg and Levy (2014) for an in-depth discussion.

HOSG is a bit slower to train than skip-gram, since \(X\) yields up to \(n_3\) times as many summands as \(X\) (but \(\ll n_3\) in practice, as \(X\) is often sparse).

4 Two Tensors for Word Embedding

As examples of useful tensors to factorize, we offer two third-order generalizations of Mikolov et al. (2013)’s context-word matrix. We are still predicting the distribution of contexts of a given word type. Our first version increases the number of parameters (giving more expressivity) by conditioning on additional information. Our second version decreases the number of parameters (giving better smoothing) by factoring the word type.

4.1 Positional Tensor

When predicting the context words in a window around a given word token, Mikolov et al. (2013) uses the same distribution to predict each of them. We propose to use different distributions at different positions in the window, via a “positional tensor”: \(X_{(\text{dog}, \text{ran}, -2)}\) is the number of times the context word \(\text{dog}\) was seen two positions to the left of \(\text{ran}\). We will predict this count using \(p(\text{dog} \mid \text{ran}, -2)\), defined from the embeddings of the word \(\text{ran}\), the position \(-2\), and the context word \(\text{dog}\) and its competitors. For a 10-word window, we have \(X \in \mathbb{R}^{V \times V \times 10}\). Considering word position should improve syntactic awareness.

4.2 Compositional Morphology Tensor

For Mikolov et al. (2013), related words such as \(\text{ran}\) and \(\text{running}\) are monolithic objects that do not share parameters. We decompose each word into a lemma \(j\) and a morphological tag \(k\). The
contexts \(i\) are still full words.\(^7\) Thus, we predict the count \(\hat{X}(d_{\text{dog}},\text{RUN},t)\) using \(p(d_{\text{dog}} \mid \text{RUN}, t)\), where \(t\) is a morphological tag such as \([\text{pos=v},\text{tense=PAST}]\).

Our model is essentially a version of the skip-gram method (Mikolov et al., 2013) that parameterizes the embedding of the word \(\text{ran}\) as a Hadamard product \(w_j \odot r_k\), where \(w_j\) embeds \(\text{RUN}\) and \(r_k\) embeds tag \(t\). This is similar to the work of Cotterell et al. (2016), who parameterized word embeddings as a sum \(w_j + r_k\) of embeddings of the component morphemes.\(^8\) Our Hadamard product embedding is in fact more general, since the additive embedding \(w_j + r_k\) can be recovered as a special case—it is equal to \((w_j; 1) \odot (1; r_k)\), which uses twice as many dimensions to embed each object.

### 5 Experiments

We build HOSG on top of the HYPERWORDS package. All models (both skip-gram and higher-order skip-gram) are trained for 10 epochs and use 5 negative samples. All models for §5.1 are trained on the Sept. 2016 dump of the full Wikipedia. All models for §5.2 were trained on the lemmatized and POS-tagged WaCky corpora (Baroni et al., 2009) for French, Italian, German and English (Joubarne and Inkpen, 2011; Leviant and Reichart, 2015). To ensure controlled and fair experiments, we follow Levy et al. (2015) for all preprocessing.

#### 5.1 Experiment 1: Positional Tensor

We postulate that the positional tensor should encode richer notions of syntax than standard bag-of-words vectors. Why? Positional information allows us to differentiate between the geometry of the cooccurrence, e.g., the is found to the left of the noun it modifies and is—more often than—close to it. Our tensor factorization model explicitly encodes this information during training.

To evaluate the vectors, we use \(\text{QVEC}\) (Tsvetkov et al., 2016), which measures Pearson’s correlation between human-annotated judgements and the vectors using CCA. The \(\text{QVEC}\) metric will be higher if the vectors better correlate with the human-annotated resource. To measure the syntactic content of the vectors, we compute the correlation between our learned vector \(w_j\) for each word and its empirical distribution \(g_i\) over universal POS tags (Petrov et al., 2012) in the UD treebank (Nivre et al., 2016). \(g_i\) can be regarded as a vector on the \((|\mathcal{T}| - 1)\)-dimensional simplex, where \(\mathcal{T}\) is the tag set. We report results on 40 languages from the UD treebanks in Table 1, using 4-word or 10-word-symmetric context windows (i.e., \(c \in \{2, 5\}\)). We find that for 77.5% of the languages, our positional tensor embeddings outperform the standard skip-gram approach on the \(\text{QVEC}\) metric.

We highlight again that the positional tensor exploits \textit{no} additional annotation, but better exploits the signal found in the raw text. Of course, our HOSG method could also be used to exploit annotations if available: e.g., one would get different embeddings by defining the relations of word \(j\) to be the labeled syntactic dependency relations in which it participates (Lin and Pantel, 2001; Levy and Goldberg, 2014a).

#### 5.2 Experiment 2: Morphology Tensor

<table>
<thead>
<tr>
<th>c = 2</th>
<th>ar</th>
<th>bg</th>
<th>ca</th>
<th>cs</th>
<th>da</th>
<th>de</th>
<th>el</th>
<th>en</th>
<th>es</th>
<th>et</th>
<th>eu</th>
<th>fa</th>
<th>fi</th>
<th>fo</th>
<th>fr</th>
<th>ga</th>
<th>gl</th>
<th>he</th>
<th>hi</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG</td>
<td>25</td>
<td>22</td>
<td>.41</td>
<td>.20</td>
<td>.21</td>
<td>.49</td>
<td>.58</td>
<td>.44</td>
<td>.61</td>
<td>.48</td>
<td>.42</td>
<td>.28</td>
<td>.46</td>
<td>.43</td>
<td>.39</td>
<td>.40</td>
<td>.10</td>
<td>.20</td>
<td>.41</td>
</tr>
<tr>
<td>HOSG</td>
<td>.40</td>
<td>.46</td>
<td>.45</td>
<td>.36</td>
<td>.50</td>
<td>.48</td>
<td>.61</td>
<td>.48</td>
<td>.42</td>
<td>.28</td>
<td>.46</td>
<td>.43</td>
<td>.39</td>
<td>.40</td>
<td>.10</td>
<td>.20</td>
<td>.41</td>
<td>.22</td>
<td>.43</td>
</tr>
<tr>
<td>Δ</td>
<td>+.15</td>
<td>+.24</td>
<td>+.14</td>
<td>+.16</td>
<td>+.29</td>
<td>+.03</td>
<td>+.04</td>
<td>+.03</td>
<td>+.19</td>
<td>+.05</td>
<td>+.04</td>
<td>+.19</td>
<td>+.05</td>
<td>+.04</td>
<td>+.03</td>
<td>+.19</td>
<td>+.05</td>
<td>+.04</td>
<td>+.19</td>
</tr>
<tr>
<td>Sr</td>
<td>.51</td>
<td>.46</td>
<td>.45</td>
<td>.35</td>
<td>.47</td>
<td>.24</td>
<td>.21</td>
<td>.24</td>
<td>.29</td>
<td>.44</td>
<td>.22</td>
<td>.28</td>
<td>.34</td>
<td>.21</td>
<td>.34</td>
<td>.15</td>
<td>.03</td>
<td>.02</td>
<td>.19</td>
</tr>
<tr>
<td>HOSG</td>
<td>.53</td>
<td>.49</td>
<td>.43</td>
<td>.46</td>
<td>.31</td>
<td>.46</td>
<td>.38</td>
<td>.45</td>
<td>.47</td>
<td>.44</td>
<td>.42</td>
<td>.46</td>
<td>.33</td>
<td>.37</td>
<td>.51</td>
<td>.58</td>
<td>.41</td>
<td>.62</td>
<td>.60</td>
</tr>
<tr>
<td>Δ</td>
<td>+.02</td>
<td>+.13</td>
<td>+.01</td>
<td>+.04</td>
<td>+.17</td>
<td>+.01</td>
<td>+.03</td>
<td>+.17</td>
<td>+.01</td>
<td>+.01</td>
<td>+.24</td>
<td>+.03</td>
<td>+.02</td>
<td>+.01</td>
<td>+.04</td>
<td>+.05</td>
<td>+.06</td>
<td>+.07</td>
<td>+.03</td>
</tr>
<tr>
<td>c = 5</td>
<td>ar</td>
<td>bg</td>
<td>ca</td>
<td>cs</td>
<td>da</td>
<td>de</td>
<td>el</td>
<td>en</td>
<td>es</td>
<td>et</td>
<td>eu</td>
<td>fa</td>
<td>fi</td>
<td>fo</td>
<td>fr</td>
<td>ga</td>
<td>gl</td>
<td>he</td>
<td>hi</td>
</tr>
<tr>
<td>SG</td>
<td>24</td>
<td>.41</td>
<td>.39</td>
<td>.29</td>
<td>.44</td>
<td>.45</td>
<td>.54</td>
<td>.52</td>
<td>.45</td>
<td>.40</td>
<td>.40</td>
<td>.38</td>
<td>.37</td>
<td>.31</td>
<td>.39</td>
<td>.54</td>
<td>.40</td>
<td>.38</td>
<td>.48</td>
</tr>
<tr>
<td>HOSG</td>
<td>29</td>
<td>.47</td>
<td>.42</td>
<td>.36</td>
<td>.49</td>
<td>.52</td>
<td>.60</td>
<td>.54</td>
<td>.48</td>
<td>.42</td>
<td>.45</td>
<td>.43</td>
<td>.44</td>
<td>.44</td>
<td>.43</td>
<td>.41</td>
<td>.42</td>
<td>.56</td>
<td>.45</td>
</tr>
<tr>
<td>Δ</td>
<td>+.05</td>
<td>+.06</td>
<td>+.03</td>
<td>+.07</td>
<td>+.04</td>
<td>+.07</td>
<td>+.06</td>
<td>+.02</td>
<td>+.03</td>
<td>+.02</td>
<td>+.05</td>
<td>+.06</td>
<td>+.06</td>
<td>+.05</td>
<td>+.01</td>
<td>+.04</td>
<td>+.05</td>
<td>+.06</td>
<td>+.07</td>
</tr>
<tr>
<td>Sr</td>
<td>.50</td>
<td>.46</td>
<td>.49</td>
<td>.42</td>
<td>.44</td>
<td>.43</td>
<td>.42</td>
<td>.44</td>
<td>.43</td>
<td>.43</td>
<td>.43</td>
<td>.43</td>
<td>.44</td>
<td>.43</td>
<td>.43</td>
<td>.43</td>
<td>.43</td>
<td>.43</td>
<td>.43</td>
</tr>
<tr>
<td>HOSG</td>
<td>.53</td>
<td>.49</td>
<td>.44</td>
<td>.50</td>
<td>.40</td>
<td>.46</td>
<td>.54</td>
<td>.50</td>
<td>.44</td>
<td>.47</td>
<td>.44</td>
<td>.43</td>
<td>.34</td>
<td>.46</td>
<td>.52</td>
<td>.58</td>
<td>.43</td>
<td>.63</td>
<td>.61</td>
</tr>
<tr>
<td>Δ</td>
<td>+.03</td>
<td>+.03</td>
<td>+.05</td>
<td>+.08</td>
<td>+.07</td>
<td>+.03</td>
<td>+.02</td>
<td>+.07</td>
<td>+.06</td>
<td>+.06</td>
<td>+.05</td>
<td>+.10</td>
<td>+.09</td>
<td>+.08</td>
<td>+.06</td>
<td>+.07</td>
<td>+.03</td>
<td>+.01</td>
<td>+.06</td>
</tr>
</tbody>
</table>

Table 1: The scores for \(\text{QVEC-CCA}\) for 40 languages. All embeddings were trained on the complete Wikipedia dump of September 2016. We measure correlation with universal POS tags from the UD treebanks.

---

\(^7\)If one wanted to extend the model to decompose the context words \(i\) as well, we see at least four approaches.

\(^8\)Cotterell et al. (2016) made two further moves that could be applied to extend the present paper. First, they allowed a word to consist of any number of (unordered) morphemes—not necessarily two—whose embeddings were combined (by summation) to get the word embedding. Second, this sum also included word-specific random noise, allowing them to learn word embeddings that deviated from compositionality.
Tensor factorization has already found uses in a few corners of NLP research. Van de Cruys et al. (2013) applied tensor factorization to model the compositional morphology of subject-verb-object triples. Similarly, Hashimoto and Tsuruoka (2015) use an implicit tensor factorization method to learn embeddings for transitive verb phrases. Tensor factorization also appears in semantic-based NLP tasks. Lei et al. (2015) explicitly factorize a tensor based on feature vectors for predicting semantic roles. Chang et al. (2014) use tensor factorization to create knowledge base embeddings optimized for relation extraction. See Bouchard et al. (2015) for a large bibliography.

Other researchers have likewise attempted to escape the bag-of-words assumption in word embeddings, e.g., Yatbaz et al. (2012) incorporates morphological and orthographic features into continuous vectors; Cotterell and Schütze (2015) consider a multi-task set-up to force morphological information into embeddings; Cotterell and Schütze (2017) jointly morphologically segment and embed words; Levy and Goldberg (2014a) derive contexts based on dependency relations; PPDB (Ganitkevitch et al., 2013) employs a mixed bag of words, parts of speech, and syntax; Rastogi et al. (2015) represent word contexts, morphology, semantic frame relations, syntactic dependency relations, and multilingual bitext counts each as separate matrices, combined via GCCA; and, finally, Schwartz et al. (2016) derived embeddings based on Hearst patterns (Hearst, 1992). Ling et al. (2015) learn position-specific word embeddings (§4.1), but do not factor them as $w_j \circ r_k$ to share parameters (we did not compare empirically to this). As demonstrated in the experiments, our tensor factorization method enables us to include other syntactic properties besides word order, e.g. morphology. Poliak et al. (2017) also create positional word embeddings. Our research direction is orthogonal to these efforts in that we provide a general purpose procedure for all sorts of higher-order cooccurrence.

## 6 Related Work

Tensor factorization has already found uses in a few corners of NLP research. Van de Cruys et al. (2013) applied tensor factorization to model the compositionality of subject-verb-object triples. Similarly, Hashimoto and Tsuruoka (2015) use an implicit tensor factorization method to learn embeddings for transitive verb phrases. Tensor factorization also appears in semantic-based NLP tasks. Lei et al. (2015) explicitly factorize a tensor based on feature vectors for predicting semantic roles. Chang et al. (2014) use tensor factorization to create knowledge base embeddings optimized for relation extraction. See Bouchard et al. (2015) for a large bibliography.

Other researchers have likewise attempted to escape the bag-of-words assumption in word embeddings, e.g., Yatbaz et al. (2012) incorporates morphological and orthographic features into continuous vectors; Cotterell and Schütze (2015) consider a multi-task set-up to force morphological information into embeddings; Cotterell and Schütze (2017) jointly morphologically segment and embed words; Levy and Goldberg (2014a) derive contexts based on dependency relations; PPDB (Ganitkevitch et al., 2013) employs a mixed bag of words, parts of speech, and syntax; Rastogi et al. (2015) represent word contexts, morphology, semantic frame relations, syntactic dependency relations, and multilingual bitext counts each as separate matrices, combined via GCCA; and, finally, Schwartz et al. (2016) derived embeddings based on Hearst patterns (Hearst, 1992). Ling et al. (2015) learn position-specific word embeddings (§4.1), but do not factor them as $w_j \circ r_k$ to share parameters (we did not compare empirically to this). As demonstrated in the experiments, our tensor factorization method enables us to include other syntactic properties besides word order, e.g. morphology. Poliak et al. (2017) also create positional word embeddings. Our research direction is orthogonal to these efforts in that we provide a general purpose procedure for all sorts of higher-order cooccurrence.

## 7 Conclusion

We have presented an interpretation of the skip-gram model as exponential family principal components analysis—a form of matrix factorization—and, thus, related it to an older strain of work. Building on this connection, we generalized the model to the tensor case. Such higher-order skip-gram methods can incorporate more linguistic structure without sacrificing scalability, as we illustrated by making our embeddings consider word order or morphology. These methods achieved better word embeddings as evaluated by standard metrics on 40 languages.

## Acknowledgements

We thank colleagues, particularly Yanif Ahmad, and anonymous reviewers for helpful discussion and feedback. The first author was supported by a DAAD Long-Term Research Grant and an NDSEG fellowship. This work was supported by the JHU Human Language Technology Center of Excellence (HLTCOE), DARPA DEFT, and NSF Grant No. 1423276. The U.S. Government is authorized to reproduce and distribute reprints for its purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.

<table>
<thead>
<tr>
<th></th>
<th>fr</th>
<th>it</th>
<th>SIML</th>
<th>RG-65</th>
<th>de</th>
<th>SIML</th>
<th>Z222</th>
<th>SG</th>
<th>HOSG</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
<td>353</td>
</tr>
<tr>
<td></td>
<td>48.3</td>
<td>43.63</td>
<td>21.33</td>
<td>44.90</td>
<td>28.39</td>
<td>50.39</td>
<td>29.75</td>
<td>70.60</td>
<td>64.50</td>
<td>64.33</td>
</tr>
</tbody>
</table>

Table 2: Word similarity results comparing the compositional morphology tensor with the standard skip-gram model. Numbers indicate Spearman’s correlation coefficient $\rho$ between human similarity judgements and the cosine distances of vectors. For each language, we compare on several sets of human judgments as listed by Faruqui et al. (2016, Table 2).

We expect these lemma embeddings to be predictive of human judgments of lemma similarity.

We evaluate using standard datasets on four languages (French, Italian, German and English). Given a list of pairs of words (always lemmata), multiple native speakers judged (on a scale of 1–10) how “similar” those words are conceptually. Our model produces a similarity judgment for each pair using the cosine similarity of their lemma embeddings $w_j$. Table 2 shows how well this learned judgment correlates with the average human judgment. Our model does achieve higher correlation than skip-gram word embeddings. Note we did not compare to a baseline that simply embeds lemmas rather than words (equivalent to fixing $r_k = 1$).

---

1. We thank colleagues, particularly Yanif Ahmad, and anonymous reviewers for helpful discussion and feedback. The first author was supported by a DAAD Long-Term Research Grant and an NDSEG fellowship. This work was supported by the JHU Human Language Technology Center of Excellence (HLTCOE), DARPA DEFT, and NSF Grant No. 1423276. The U.S. Government is authorized to reproduce and distribute reprints for its purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA or the U.S. Government.
References


Dekang Lin and Patrick Pantel. 2001. Dirt @sbt@discovery of inference rules from text. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’01, pages 323–328, New York, NY, USA. ACM.


