# A Graph Auto-encoder Model of Derivational Morphology

**Valentin Hofmann**[*‡], **Hinrich Schütze**[‡], **Janet B. Pierrehumbert**[†*]

[*]Faculty of Linguistics, University of Oxford
[†]Department of Engineering Science, University of Oxford
[‡]Center for Information and Language Processing, LMU Munich
valentin.hofmann@ling-phil.ox.ac.uk

## Abstract

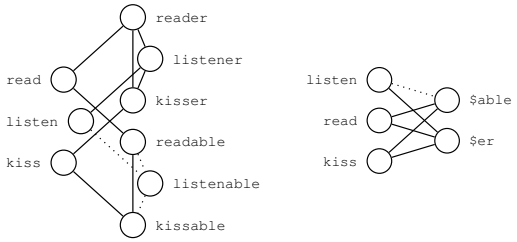
There has been little work on modeling the morphological well-formedness (MWF) of derivatives, a problem judged to be complex and difficult in linguistics (Bauer, 2019). We present a graph auto-encoder that learns embeddings capturing information about the compatibility of affixes and stems in derivation. The auto-encoder models MWF in English surprisingly well by combining syntactic and semantic information with associative information from the mental lexicon.

## 1 Introduction

A central goal of morphology is, as famously put by Aronoff (1976), "to tell us what sort of new words a speaker can form." This definition is tightly intertwined with the notion of morphological well-formedness (MWF). While non-existing morphologically well-formed words such as pro$computer$ism conform to the morphological patterns of a language and could be formed, non-existing morphologically ill-formed words such as pro$and$ism violate the patterns and are deemed impossible (Allen, 1979).

More recent research has shown that MWF is a gradient rather than binary property: non-existing words that conform to the morphological patterns of a language differ in how likely they are to be actually created by speakers (Pierrehumbert, 2012). This is particularly true in the case of derivational morphology, which is not obligatory and often serves communicative needs (Bauer, 2019). As a result, the degree of MWF of a non-existing derivative is influenced by a multitude of factors and judged to be hard to predict (Bauer, 2001).

In NLP, the lack of reliable ways to estimate the MWF of derivatives poses a bottleneck for generative models, particularly in languages exhibiting a rich derivational morphology; e.g., while inflected forms can be translated by generating morphologically corresponding forms in the target language (Minkov et al., 2007), generating derivatives is still a major challenge for machine translation systems (Sreelekha and Bhattacharyya, 2018). Similar problems exist in the area of automatic language generation (Gatt and Krahmer, 2018).




(a) Mental lexicon ($\mathcal{L}$) (b) Two-mode projection ($\mathcal{B}$)

Figure 1: Derivatives in the mental lexicon $\mathcal{L}$ (a) and the derivational graph (DG), their derivational projection $\mathcal{B}$ (b). Predicting whether a word is part of a derivational abstraction corresponds to predicting a single edge in the DG (dotted lines).

This study takes a first step towards computationally modeling the MWF of English derivatives. We present a derivational graph auto-encoder (DGA) that combines semantic and syntactic information with associative information from the mental lexicon, achieving very good results on MWF prediction and performing on par with a character-based LSTM at a fraction of the number of trainable parameters. The model produces embeddings that capture information about the compatibility of affixes and stems in derivation and can be used as pretrained input to other NLP applications.[1]

## 2 Derivational Morphology

### 2.1 Inflection and Derivation

Linguistics divides morphology into inflection and derivation. While inflection refers to the different

[1]We make all our code and data publicly available at https://github.com/valentinhofmann/dga.

word forms of a lexeme, e.g., `listen`, `listens`, and `listened`, derivation refers to the different lexemes of a word family, e.g., `listen`, `listener`, and `listenable`. There are several differences between inflection and derivation, some of which are highly relevant for NLP.

Firstly, while inflection is obligatory and determined by syntactic needs, the existence of derivatives is mainly driven by communicative goals, allowing to express a varied spectrum of meanings (Acquaviva, 2016). Secondly, derivation can produce a larger number of new words than inflection since it is iterable (Haspelmath and Sims, 2010); derivational affixes can be combined, in some cases even recursively (e.g., `post$post$modern$ism`). However, morphotactic constraints restrict the ways in which affixes can be attached to stems and other affixes (Hay and Plag, 2004); e.g., the suffix `$less` can be combined with `$ness` (`atom$less$ness`) but not with `$ity` (`atom$less$ity`).

The semantic and formal complexity of derivation makes predicting the MWF of derivatives more challenging than the MWF of inflectional forms (Anshen and Aronoff, 1999; Bauer, 2019). Here, we model the MWF of derivatives as the likelihood of their existence in the mental lexicon.

### 2.2 Derivatives in the Mental Lexicon

How likely a derivative is to exist is influenced by various factors (Bauer, 2001; Pierrehumbert and Granell, 2018). In this study, we concentrate on the role of the structure of the mental lexicon.

The mental lexicon can be thought of as a set of associations between meaning $m$ and form $f$, i.e., words, organized in a network, where links correspond to shared semantic and phonological properties (see Pierrehumbert (2012) for a review). Since we base our study on textual data, we will treat the form of words orthographically rather than phonologically. We will refer to the type of information conveyed by the cognitive structure of the mental lexicon as *associative* information.

Sets of words with similar semantic and formal properties form clusters in the mental lexicon (Alegre and Gordon, 1999). The semantic and formal properties reinforced by such clusters create abstractions that can be extended to new words (Bybee, 1995). If the abstraction hinges upon a shared derivational pattern, the effect of such an extension is a new derivative. The extent to which a word conforms to the properties of the cluster influences how likely the abstraction (in our case a derivational pattern) is to be extended to that word. This is what is captured by the notion of MWF.

### 2.3 Derivational Graphs

The main goal of this paper is to predict the MWF of morphological derivatives (i.e., how likely is a word to be formed as an extension of a lexical cluster) by directly leveraging associative information. Since links in the mental lexicon reflect semantic and formal similarities of various sorts, many of which are not morphological (Tamariz, 2008), we want to create a distilled model of the mental lexicon that only contains derivational information. One way to achieve this is by means of a derivational projection of the mental lexicon, a network that we call the *Derivational Graph* (DG).

Let $\mathcal{L} = (\mathcal{W}, \mathcal{Q})$ be a graph of the mental lexicon consisting of a set of words $\mathcal{W}$ and a set of links between the words $\mathcal{Q}$. Let $\mathcal{W}_a \subset \mathcal{W}$ be a set of words forming a fully interconnected cluster in $\mathcal{L}$ due to a shared derivational pattern $a$. We define $\mathcal{S}_a$ as the set of stems resulting from stripping off $a$ from the words in $\mathcal{W}_a$ and $\mathcal{R}_a = \{(s, a)\}_{s \in \mathcal{S}_a}$ as the corresponding set of edges between the stems and the shared derivational pattern. We then define the two-mode derivational projection $\mathcal{B}$ of $\mathcal{L}$ as the Derivational Graph (DG) where $\mathcal{B} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \bigcup_a (\mathcal{S}_a \cup \{a\})$ and $\mathcal{E} = \bigcup_a \mathcal{R}_a$. Figure 1 gives an example of $\mathcal{L}$ and DG (= $\mathcal{B}$).

The DG is a bipartite graph whose nodes consist of stems $s \in \mathcal{S}$ with $\mathcal{S} = \bigcup_a \mathcal{S}_a$ and derivational patterns $a \in \mathcal{A}$ with $\mathcal{A} = \bigcup_a \{a\}$. The derivational patterns are sequences of affixes such as `re$$ize$ate$ion` in the case of `revitalization`. The cognitive plausibility of this setup is supported by findings that affix groups can trigger derivational generalizations in the same way as individual affixes (Stump, 2017, 2019).

We define $\mathbf{B} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ to be the adjacency matrix of $\mathcal{B}$. The degree of an individual node $n$ is $d(n)$. We further define $\Gamma^1(n)$ as the set of one-hop neighbors and $\Gamma^2(n)$ as the set of two-hop neighbors of $n$. Notice that $\Gamma^1(s) \subseteq \mathcal{A}$, $\Gamma^1(a) \subseteq \mathcal{S}$, $\Gamma^2(s) \subseteq \mathcal{S}$, and $\Gamma^2(a) \subseteq \mathcal{A}$ for any $s$ and $a$ since the DG is bipartite.

The advantage of this setup of DGs is that it abstracts away information not relevant to derivational morphology while still allowing to interpret results in the light of the mental lexicon. The cre-

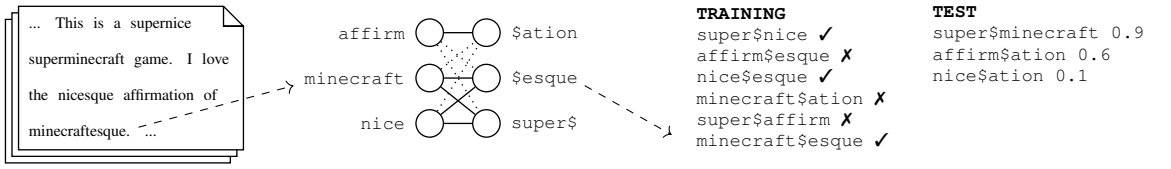



Figure 2: Experimental setup. We extract DGs from Reddit and train link prediction models on them. In the shown toy example, the derivatives `super$minecraft` and `affirm$ation` are held out for the test set.

ation of a derivative corresponds to a new link between a stem and a derivational pattern in the DG, which in turn reflects the inclusion of a new word into a lexical cluster with a shared derivational pattern in the mental lexicon.

## 3 Experimental Data

### 3.1 Corpus

We base our study on data from the social media platform Reddit.[2] Reddit is divided into so-called *subreddits* (SRs), smaller communities centered around shared interests. SRs have been shown to exhibit community-specific linguistic properties (del Tredici and Fernández, 2018).

We draw upon the Baumgartner Reddit Corpus, a collection of publicly available comments posted on Reddit since 2005.[3] The preprocessing of the data is described in Appendix A.1. We examine data in the SRs r/cfb (cfb – college football), r/gaming (gam), r/leagueoflegends (lol), r/movies (mov), r/nba (nba), r/nfl (nfl), r/politics (pol), r/science (sci), and r/technology (tec) between 2007 and 2018. These SRs were chosen because they are of comparable size and are among the largest SRs (see Table 1). They reflect three distinct areas of interest, i.e., sports (cfb, nba, nfl), entertainment (gam, lol, mov), and knowledge (pol, sci, tec), thus allowing for a multifaceted view on how topical factors impact MWF: seeing MWF as an emergent property of the mental lexicon entails that communities with different lexica should differ in what derivatives are most likely to be created.

### 3.2 Morphological Segmentation

Many morphologically complex words are not decomposed into their morphemes during cognitive processing (Sonnenstuhl and Huth, 2002). Based on experimental findings in Hay (2001), we segment a morphologically complex word only if the stem has a higher token frequency than the derivative (in a given SR). Segmentation is performed by means of an iterative affix-stripping algorithm introduced in Hofmann et al. (2020) that is based on a representative list of productive prefixes and suffixes in English (Crystal, 1997). The algorithm is sensitive to most morpho-orthographic rules of English (Plag, 2003): when `$ness` is removed from `happi$ness`, e.g., the result is `happy`, not `happi`. See Appendix A.2. for details.

The segmented texts are then used to create DGs as described in Section 2.3. All processing is done separately for each SR, i.e., we create a total of nine different DGs. Figure 2 illustrates the general experimental setup of our study.

| SR | $n_w$ | $n_t$ | $\lvert\mathcal{S}\rvert$ | $\lvert\mathcal{A}\rvert$ | $\lvert\mathcal{E}\rvert$ |
|---|---|---|---|---|---|
| cfb | 475,870,562 | 522,675 | 10,934 | 2,261 | 46,110 |
| nba | 898,483,442 | 801,260 | 13,576 | 3,023 | 64,274 |
| nfl | 911,001,246 | 791,352 | 13,982 | 3,016 | 64,821 |
| gam | 1,119,096,999 | 1,428,149 | 19,306 | 4,519 | 107,126 |
| lol | 1,538,655,464 | 1,444,976 | 18,375 | 4,515 | 104,731 |
| mov | 738,365,964 | 860,263 | 15,740 | 3,614 | 77,925 |
| pol | 2,970,509,554 | 1,576,998 | 24,175 | 6,188 | 143,880 |
| sci | 277,568,720 | 528,223 | 11,267 | 3,323 | 58,290 |
| tec | 505,966,695 | 632,940 | 11,986 | 3,280 | 63,839 |

Table 1: SR statistics. $n_w$: number of tokens; $n_t$: number of types; $|\mathcal{S}|$: number of stem nodes; $|\mathcal{A}|$: number of affix group nodes; $|\mathcal{E}|$: number of edges.

## 4 Models

Let $W$ be a Bernoulli random variable denoting the property of being morphologically well-formed. We want to model $P(W|d, C_r) = P(W|s, a, C_r)$, i.e., the probability that a derivative $d$ consisting of stem $s$ and affix group $a$ is morphologically well-formed according to SR corpus $C_r$.

Given the established properties of derivational morphology (see Section 2), a good model of $P(W|d, C_r)$ should include both semantics and formal structure,

$$P(W|d, C_r) = P(W|m_s, f_s, m_a, f_a, C_r), \quad (1)$$

where $m_s$, $f_s$, $m_a$, $f_a$, are meaning and form (here

[2] `reddit.com`
[3] `files.pushshift.io/reddit/comments`

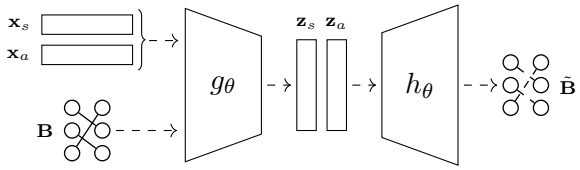



Figure 3: DGA model architecture. The DGA takes as input an adjacency matrix $\mathbf{B}$ and additional feature vectors $\mathbf{x}_s$ and $\mathbf{x}_a$ and learns embeddings $\mathbf{z}_s$ and $\mathbf{z}_a$.

modeled orthographically, see Section 2.2) of the involved stem and affix group, respectively. The models we examine in this study vary in which of these features are used, and how they are used.

### 4.1 Derivational Graph Auto-encoder

We model $P(W|d, C_r)$ by training a graph auto-encoder (Kipf and Welling, 2016, 2017) on the DG $\mathcal{B}$ of each SR. The graph auto-encoder attempts to reconstruct the adjacency matrix $\mathbf{B}$ (Section 2.3) of the DG by means of an encoder function $g_\theta$ and a decoder function $h_\theta$, i.e., its basic structure is

$$\tilde{\mathbf{B}} = h_\theta\left(g_\theta\left(\mathbf{B}\right)\right), \tag{2}$$

where $\tilde{\mathbf{B}}$ is the reconstructed version of $\mathbf{B}$. The specific architecture we use (see Figure 3), which we call a *Derivational Graph Auto-encoder* (DGA), is a variation of the bipartite graph auto-encoder (van den Berg et al., 2018).

**Encoder.** The encoder $g_\theta$ takes as one of its inputs the adjacency matrix $\mathbf{B}$ of the DG $\mathcal{B}$. This means we model $f_s$ and $f_a$, the stem and affix group forms, by means of the associative relationships they create in the mental lexicon. Since a DG has no information about semantic relationships between nodes within $\mathcal{S}$ and $\mathcal{A}$, we reintroduce meaning as additional feature vectors $\mathbf{x}_s, \mathbf{x}_a \in \mathbb{R}^n$ for $m_s$ and $m_a$, stem and affix group embeddings that are trained separately on the SR texts. The input to $g_\theta$ is thus designed to provide complementary information: associative information ($\mathbf{B}$) and semantic information ($\mathbf{x}_s$ and $\mathbf{x}_a$).

For the encoder to be able to combine the two types of input in a meaningful way, the choice of $g_\theta$ is crucial. We model $g_\theta$ as a graph convolutional network (Kipf and Welling, 2016, 2017), providing an intuitive way to combine information from the DG with additional information. The graph convolutional network consists of $L$ convolutional layers. Each layer (except for the last one) performs two steps: message passing and activation.

During the message passing step (Dai et al., 2016; Gilmer et al., 2017), transformed versions of the embeddings $\mathbf{x}_s$ and $\mathbf{x}_a$ are sent along the edges of the DG, weighted, and accumulated. We define $\Gamma^1_+(s) = \Gamma^1(s) \cup \{s\}$ as the set of nodes whose transformed embeddings are weighted and accumulated for a particular stem $s$. $\Gamma^1_+(s)$ is extracted from the adjacency matrix $\mathbf{B}$ and consists of the one-hop neighbors of $s$ and $s$ itself. The message passing propagation rule (Kipf and Welling, 2016, 2017) can then be written as

$$\mathbf{m}_s^{(l)} = \sum_{n \in \Gamma^1_+(s)} \frac{\mathbf{x}_n^{(l-1)} \mathbf{W}^{(l)}}{\sqrt{|\Gamma^1_+(s)||\Gamma^1_+(n)|}}, \tag{3}$$

where $\mathbf{W}^{(l)}$ is the trainable weight matrix of layer $l$, $\mathbf{x}_n^{(l-1)}$ is the embedding of node $n$ from layer $l-1$ with $\mathbf{x}_n^{(0)} = \mathbf{x}_n$, and $\sqrt{|\Gamma^1_+(s)||\Gamma^1_+(n)|}$ is the weighting factor. The message passing step is performed analogously for affix groups. The matrix form of Equation 3 is given in Appendix A.3.

Intuitively, a message passing step takes embeddings of all neighbors of a node and the embedding of the node itself, transforms them, and accumulates them by a normalized sum. Given that the DG $\mathcal{B}$ is bipartite, this means for a stem $s$ that the normalized sum contains $d(s)$ affix group embeddings and one stem embedding (and analogously for affix groups). The total number of convolutional layers $L$ determines how far the influence of a node can reach. While one convolution allows nodes to receive information from their one-hop neighbors (stems from affix groups they co-occur with and vice versa), two convolutions add information from the two-hop neighbors (stems from stems co-occurring with the same affix group and vice versa), etc. (see Figure 4).

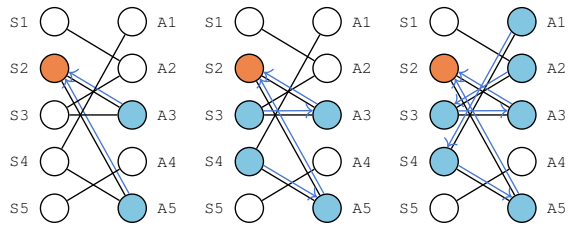
During the activation step, the output of the convolutional layer $l$ for a particular stem $s$ is

$$\mathbf{x}_s^{(l)} = \mathrm{ReLU}\left(\mathbf{m}_s^{(l)}\right), \tag{4}$$

where $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ is a rectified linear unit (Nair and Hinton, 2010). The final output of the encoder is

$$\mathbf{z}_s = \mathbf{m}_s^{(L)}, \tag{5}$$

i.e., there is no activation in the last layer. The activation step is again performed analogously for affix groups. $\mathbf{z}_s$ and $\mathbf{z}_a$ are representations of $s$ and $a$ enriched with information about the semantics of nodes in their DG neighborhood.




(a) $L = 1$ (b) $L = 2$ (c) $L = 3$

Figure 4: Influence of $L$, the number of convolutional layers, on message passing. The blue nodes illustrate neighbors whose messages can be received by the orange node under varying $L$.

**Decoder.** We model the decoder as a simple bilinear function,

$$h_\theta\left(\mathbf{z}_s, \mathbf{z}_a\right) = \sigma\left(\mathbf{z}_s^\top \mathbf{z}_a\right), \tag{6}$$

where $\sigma$ is the sigmoid and $\mathbf{z}_s$ and $\mathbf{z}_a$ are the outputs of the encoder.[4] We set $P(W|d, C_r) = h_\theta\left(\mathbf{z}_s, \mathbf{z}_a\right)$ and interpret this as the probability that the corresponding edge in a DG constructed from a corpus drawn from the underlying distribution exists. The resulting matrix $\tilde{\mathbf{B}}$ in Equation 2 is then the reconstructed adjacency matrix of DG.

Notice that the only trainable parameters of the DGA are the weight matrices $\mathbf{W}^{(l)}$. To put the performance of the DGA into perspective, we compare against four baselines, which we present in decreasing order of sophistication.

### 4.2 Baseline 1: Character-based Model (CM)

We model $P(W|d, C_r)$ as $P(W|f_s, f_a, C_r)$ using a character-based model (CM), i.e., as opposed to the DGA, $f_s$ and $f_a$ are modeled directly by means of their orthographic form. This provides the CM with phonological information, a central predictor of MWF (see Section 2.2). CM might also learn semantic information during training, but it is not directly provided with it. Character-based models show competitive results on derivational tasks (Cotterell et al., 2017; Vylomova et al., 2017; Deutsch et al., 2018), a good reason to test their performance on MWF prediction.

We use two one-layer bidirectional LSTMs to encode the stem and affix group into a vector $\mathbf{o}$ by concatenating the last hidden states from both LSTM directions $\overrightarrow{\mathbf{h}}_s$, $\overleftarrow{\mathbf{h}}_s$, $\overrightarrow{\mathbf{h}}_a$, and $\overleftarrow{\mathbf{h}}_a$,

$$\mathbf{o} = [\overrightarrow{\mathbf{h}}_s \oplus \overleftarrow{\mathbf{h}}_s \oplus \overrightarrow{\mathbf{h}}_a \oplus \overleftarrow{\mathbf{h}}_a], \tag{7}$$

where $\oplus$ denotes concatentation. $\mathbf{o}$ is then fed into a two layer feed-forward neural network with a $\mathrm{ReLU}$ non-linearity after the first layer.[5] The activation function after the second layer is $\sigma$.

### 4.3 Baseline 2: Neural Classifier (NC)

We model $P(W|d, C_r)$ as $P(W|m_s, m_a, C_r)$ using a neural classifier (NC) whose architecture is similar to the auto-encoder setup of the DGA.

Similarly to the DGA, $m_s$ and $m_a$ are modeled by means of stem and affix group embeddings trained separately on the SRs. The first encoder-like part of the NC is a two-layer feed-forward neural network with a $\mathrm{ReLU}$ non-linearity after the first layer. The second decoder-like part of the NC is an inner-product layer as in the DGA. Thus, the NC is identical to the DGA except that it does not use associative information from the DG via a graph convolutional network; it only has information about the stem and affix group meanings.

### 4.4 Baseline 3: Jaccard Similarity (JS)

We model $P(W|d, C_r)$ as $P(W|f_s, f_a, C_r)$. Like in the DGA, we model the stem and affix group forms by means of the associative relationships they create in the mental lexicon. Specifically, we predict links without semantic information.

In feature-based machine learning, link prediction is performed by defining similarity measures on a graph and ranking node pairs according to these features (Liben-Nowell and Kleinberg, 2003). We apply four common measures, most of which have to be modified to accommodate the properties of bipartite DGs. Here, we only cover the best performing measure, Jaccard similarity (JS). JS is one of the simplest graph-based similarity measures, so it is a natural baseline for answering the question: how far does simple graph-based similarity get you at predicting MWF? See Appendix A.4 for the other three measures.

The JS score of an edge $(s, a)$ is traditionally defined as

$$\zeta_{JS}(s, a) = \frac{|\Gamma^1(s) \cap \Gamma^1(a)|}{|\Gamma^1(s) \cup \Gamma^1(a)|}. \tag{8}$$

However, since $\Gamma^1(s) \cap \Gamma^1(a) = \emptyset$ for any $s$ and $a$ (the DG is bipartite), we redefine the set of common neighbors of two nodes $n$ and $m$, $\Gamma_\cap(n, m)$, as $\Gamma^2(n) \cap \Gamma^1(m)$, i.e., the intersection of the two-hop neighbors of $n$ and the one-hop neighbors of

[4] Besides the simple dot-product decoder, we also implemented a bilinear decoder with $h(\mathbf{z}_s, \mathbf{z}_a) = \sigma(\mathbf{z}_s^\top \mathbf{Q} \mathbf{z}_a)$, where $\mathbf{Q}$ is a trainable weight matrix. However, the model performed significantly worse.

[5] We also experimented with only one layer, but it performed considerably worse.

$m$, and analogously $\Gamma_{\cup}(n, m)$ as $\Gamma^2(n) \cup \Gamma^1(m)$. Since these are asymmetric definitions, we define

$$\zeta_{JS}(s, a) = \frac{|\Gamma_{\cap}(s, a)|}{|\Gamma_{\cup}(s, a)|} + \frac{|\Gamma_{\cap}(a, s)|}{|\Gamma_{\cup}(a, s)|} \quad (9)$$

JS assumes that a stem that is already similar to a lexical cluster in its derivational patterns is more likely to become even more similar to the cluster than a less similar stem.

### 4.5 Baseline 4: Bigram Model (BM)

We again model $P(W|d, C_r)$ as $P(W|f_s, f_a, C_r)$, leaving aside semantic information. However, in contrast to JS, this model implements the classic approach of Fabb (1988), according to which pairwise constraints on affix combinations, or combinations of a stem and an affix, determine the allowable sequences. Taking into account more recent results on morphological gradience, we do not model these selection restrictions with binary rules. Instead, we use transition probabilities, beginning with the POS of the stem $s$ and working outwards to each following suffix $a^{(s)}$ or preceding prefix $a^{(p)}$. Using a simple bigram model (BM), we can thus calculate the MWF of a derivative as

$$P(W|d, C_r) = P(a^{(s)}|s) \cdot P(a^{(p)}|s), \quad (10)$$

where $P(a^{(s)}|s) = P(a_1^{(s)}|s) \prod_{i=2}^{n} P(a_i^{(s)}|a_{i-1}^{(s)})$ is the probability of the suffix group conditioned on the POS of the stem. $P(a^{(p)}|s)$ is defined analogously for prefix groups.

## 5 Experiments

### 5.1 Setup

We train all models on the nine SRs using the same split of $\mathcal{E}$ into training ($n_{train}^{(p)} = 0.85 \cdot |\mathcal{E}|$), validation ($n_{val}^{(p)} = 0.05 \cdot |\mathcal{E}|$), and test ($n_{test}^{(p)} = 0.1 \cdot |\mathcal{E}|$) edges. For validation and test, we randomly sample $n_{val}^{(n)} = n_{val}^{(p)}$ and $n_{test}^{(n)} = n_{test}^{(p)}$ non-edges $(s, a) \notin \mathcal{E}$ as negative examples such that both sets are balanced (0.5 positive, 0.5 negative).

For training, we sample $n_{train}^{(n)} = n_{train}^{(p)}$ non-edges $(s, a) \notin \mathcal{E}$ in every epoch (i.e., the set of sampled non-edges changes in every epoch). Nodes are sampled according to their degree with $P(n) \propto d(n)$, a common strategy in bipartite link prediction (Chen et al., 2017). We make sure non-edges sampled in training are not in the validation or test sets. During the test phase, we rank all edges according to their predicted scores.

| Model | $n_p$ |
|---|---|
| DGA+ | 30,200 |
| DGA | 20,200 |
| CM | 349,301 |
| NC+ | 30,200 |
| NC | 20,200 |

Table 2: Number of trainable parameters for neural models. $n_p$: number of trainable parameters.

We evaluate the models using average precision (AP) and area under the ROC curve (AUC), two common evaluation measures in link prediction that do not require a decision threshold. AP emphasizes the correctness of the top-ranked edges (Su et al., 2015) more than AUC.

### 5.2 Training Details

**DGA, DGA+:** We use binary cross entropy as loss function. Hyperparameter tuning is performed on the validation set. We train the DGA for 600 epochs using Adam (Kingma and Ba, 2015) with a learning rate of 0.01.[6] We use $L = 2$ hidden layers in the DGA with a dimension of 100. For regularization, we apply dropout of 0.1 after the input layer and 0.7 after the hidden layers. For $\mathbf{x}_s$ and $\mathbf{x}_a$, we use 100-dimensional GloVe embeddings (Pennington et al., 2014) trained on the segmented text of the individual SRs with a window size of 10. These can be seen as GloVe variants of traditional morpheme embeddings as proposed, e.g., by Qiu et al. (2014), with the sole difference that we use affix groups instead of individual affixes. For training the embeddings, derivatives are segmented into prefix group, stem, and suffix group. In the case of both prefix and suffix groups, we add prefix and suffix group embeddings.

Since the window size impacts the information represented by the embeddings, with larger windows tending to capture topical and smaller windows morphosyntactic information (Lison and Kutuzov, 2017), we also train the DGA with 200-dimensional embeddings consisting of concatenated 100-dimensional embeddings trained with window sizes of 10 and 1, respectively (DGA+).[7] Since DGA already receives associative informa-

[6] The number of epochs until convergence lies within the typical range of values for graph convolutional networks.

[7] We experimented with using vectors trained on isolated pairs of stems and affix groups instead of window-1 vectors trained on the full text, but the performance was comparable. We also implemented the DGA using only window-1 vectors (without concatenating them with window-10 vectors), but it performed considerably worse.

| | sports | | | | | | entertainment | | | | | | knowledge | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cfb | | nba | | nfl | | gam | | lol | | mov | | pol | | sci | | tec | | $\mu \pm \sigma$ | |
| Model | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC |
| DGA+ | .783 | .754 | .764 | .749 | .773 | .751 | .775 | .759 | .758 | .740 | .772 | .749 | .777 | .766 | .809 | .795 | .799 | .778 | .779±.015 | .760±.016 |
| DGA | .760 | .730 | .754 | .731 | .762 | .740 | .762 | .743 | .752 | .738 | .765 | .747 | .764 | .750 | .783 | .770 | .781 | .761 | .765±.010 | .746±.012 |
| CM | .745 | .745 | .751 | .759 | .764 | .766 | .768 | .776 | .766 | .773 | .769 | .780 | .776 | .786 | .793 | .804 | .768 | .775 | .767±.013 | .774±.016 |
| NC+ | .737 | .739 | .729 | .733 | .737 | .740 | .722 | .728 | .730 | .733 | .732 | .741 | .725 | .731 | .772 | .781 | .758 | .756 | .738±.016 | .742±.016 |
| NC | .704 | .710 | .705 | .715 | .719 | .728 | .709 | .714 | .699 | .711 | .709 | .720 | .695 | .708 | .731 | .743 | .734 | .737 | .712±.013 | .721±.012 |
| JS | .632 | .593 | .617 | .582 | .626 | .588 | .619 | .588 | .609 | .584 | .622 | .589 | .614 | .591 | .649 | .617 | .638 | .608 | .625±.012 | .593±.011 |
| BM | .598 | .602 | .592 | .597 | .600 | .600 | .592 | .592 | .583 | .585 | .596 | .594 | .583 | .584 | .610 | .601 | .589 | .596 | .594±.008 | .595±.006 |

Table 3: Performance on MWF prediction. The table shows AP and AUC of the models for the nine SRs as well as averaged scores. Grey highlighting illustrates the best score in a column, light grey the second-best.

tion from the DG and semantic information from the embeddings trained with window size 10, the main advantage of DGA+ should lie in additional syntactic information.

**CM:** We use binary cross entropy as loss function. We train the CM for 20 epochs using Adam with a learning rate of 0.001. Both input character embeddings and hidden states of the bidirectional LSTMs have 100 dimensions. The output of the first feed-forward layer has 50 dimensions. We apply dropout of 0.2 after the embedding layer as well as the first feed-forward layer.

**NC, NC+:** All hyperparameters are identical to the DGA and the DGA+, respectively.

**JS:** Similarity scores are computed on the SR training sets.

**BM:** Transition probabilities are maximum likelihood estimates from the SR training sets. If a stem is assigned several POS tags by the tagger, we take the most frequent one.

Table 2 summarizes the number of trainable parameters for the neural models. Notice that CM has more than 10 times as many trainable parameters as DGA+, DGA, NC+, and NC.

### 5.3 Results

The overall best performing models are DGA+ and CM (see Table 3). While DGA+ beats CM on all SRs except for lol in AP, CM beats DGA+ on all SRs except for cfb and tec in AUC. Except for CM, DGA+ beats all other models on all SRs in both AP and AUC, i.e., it is always the best or second-best model. DGA beats all models except for DGA+ and CM on all SRs in AP but has lower AUC than NC+ on three SRs. It also outperforms CM on three SRs in AP. NC+ and NC mostly have scores above 0.7, showing that traditional morpheme embeddings also capture information about the compatibility of affixes and stems (albeit to a lesser degree than models with associative or orthographic information). Among the non-neural methods, JS outperforms BM (and the other non-neural link prediction models, see Appendix A.4) in AP, but is beaten by BM in AUC on six SRs.

The fact that DGA+ performs on par with CM while using less than 10% of CM's parameters demonstrates the power of incorporating associative information from the mental lexicon in modeling the MWF of derivatives. This result is even more striking since DGA+, as opposed to CM, has no direct access to orthographic (i.e., phonological) information. At the same time, CM's high performance indicates that orthographic information is an important predictor of MWF.

## 6 Derivational Embeddings

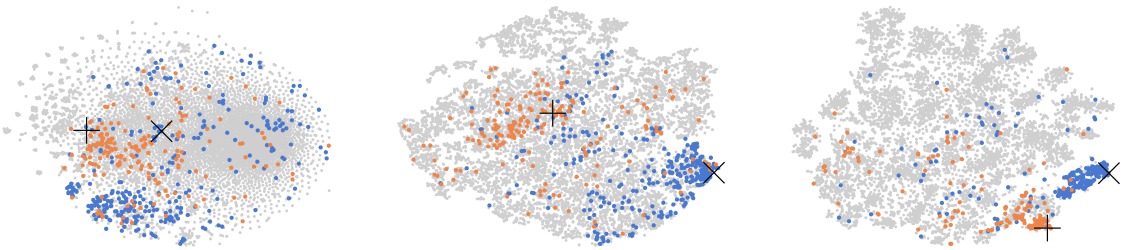
### 6.1 Comparison with Input Vectors

To understand better how associative information from the DG increases performance, we examine how DGA+ changes the shape of the vector space by comparing input vs. learned embeddings ($\mathbf{X}$ vs. $\mathbf{Z}_{DGA+}$), and contrast that with NC+ ($\mathbf{X}$ vs. $\mathbf{Z}_{NC+}$). A priori, there are two opposing demands the embeddings need to respond to: (i) as holds for bipartite graphs in general (Gao et al., 2018), the two node sets (stems and affix groups) should form two separated clusters in embedding space; (ii) stems associated with the same affix group should form clusters in embedding space that are close to the embedding of the respective affix group.

For this analysis, we define $\delta(\mathcal{N}, \mathbf{v})$ as the mean cosine similarity between the embeddings of a node set $\mathcal{N}$ and an individual embedding $\mathbf{v}$,

$$\delta(\mathcal{N}, \mathbf{v}) = \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \cos\left(\mathbf{u}_n, \mathbf{v}\right), \qquad (11)$$

where $\mathbf{u}_n$ is the embedding of node $n$. We calculate $\delta$ for the set of stem nodes $\mathcal{S}$ and their centroid $\mathbf{c}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathbf{u}_s$ as well as the set of affix group



(a) $\mathbf{X}$ (b) $\mathbf{Z}_{NC+}$ (c) $\mathbf{Z}_{DGA+}$

Figure 5: Comparison of input embeddings $\mathbf{X}$ with learned representations $\mathbf{Z}_{NC+}$ and $\mathbf{Z}_{DGA+}$. The plots are t-SNE projections (van der Maaten and Hinton, 2008) of the embedding spaces. We highlight two example sets of stems occurring with a common affix: the blue points are stems occurring with `$esque`, the orange points stems occurring with `$ful`. $\times$ marks the embedding of `$esque`, $+$ the embedding of `$ful`.

| Measure | $\mathbf{X}$ | $\mathbf{Z}_{NC+}$ | $\mathbf{Z}_{DGA+}$ |
|---|---|---|---|
| $\delta(\mathcal{S}, \mathbf{c}_{\mathcal{S}})$ | .256 ± .026 | .500 ± .027 | .487 ± .022 |
| $\delta(\mathcal{A}, \mathbf{c}_{\mathcal{A}})$ | .377 ± .017 | .522 ± .016 | .322 ± .030 |
| $\delta(\mathcal{S}_a, \mathbf{c}_{\mathcal{S}_a})$ | .281 ± .006 | .615 ± .017 | .671 ± .024 |
| $\delta(\mathcal{S}_a, \mathbf{u}_a)$ | .133 ± .006 | .261 ± .022 | .278 ± .033 |

Table 4: Comparison of $\mathbf{X}$ with $\mathbf{Z}_{NC+}$ and $\mathbf{Z}_{DGA+}$. The table shows topological measures highlighting differences between the input and learned embeddings.

nodes $\mathcal{A}$ and their centroid $\mathbf{c}_{\mathcal{A}} = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}} \mathbf{u}_a$. Table 4 shows that while NC+ makes the embeddings of both $\mathcal{S}$ and $\mathcal{A}$ more compact (higher similarity in $\mathbf{Z}_{NC+}$ than in $\mathbf{X}$), DGA+ makes $\mathcal{S}$ more compact, too, but decreases the compactness of $\mathcal{A}$ (lower similarity in $\mathbf{Z}_{DGA+}$ than in $\mathbf{X}$). $\mathbf{Z}_{NC+}$ meets (i) to a greater extent than $\mathbf{Z}_{DGA+}$.

We then calculate $\delta$ for all sets of stems $\mathcal{S}_a$ occurring with a common affix group $a$ and their centroids $\mathbf{c}_{\mathcal{S}_a} = \frac{1}{|\mathcal{S}_a|}\sum_{s\in\mathcal{S}_a} \mathbf{u}_s$. We also compute $\delta$ for all $\mathcal{S}_a$ and the embeddings of the corresponding affix groups $\mathbf{u}_a$. As Table 4 shows, both values are much higher in $\mathbf{Z}_{DGA+}$ than in $\mathbf{X}$, i.e., DGA+ brings stems with a common affix group $a$ (lexical clusters in the mental lexicon) close to each other while at the same time moving $a$ into the direction of the stems. The embeddings $\mathbf{Z}_{NC+}$ exhibit a similar pattern, but more weakly than $\mathbf{Z}_{DGA+}$ (see Table 4 and Figure 5). $\mathbf{Z}_{DGA+}$ meets (ii) to a greater extent than $\mathbf{Z}_{NC+}$.

Thus, DGA+ and NC+ solve the tension between (i) and (ii) differently; the associative information from the mental lexicon allows DGA+ to put a greater emphasis on (ii), leading to higher performance in MWF prediction.

### 6.2 Comparison between SRs

Another reason for the higher performance of the models with associative information could be that their embeddings capture differences in derivational patterns between the SR communities. To examine this hypothesis, we map the embeddings $\mathbf{Z}_{DGA+}$ of all SRs into a common vector space by means of orthogonal procrustes alignment (Schönemann, 1966), i.e., we optimize

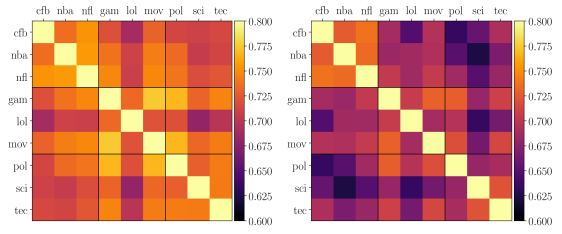
$$\mathbf{R}^{(i)} = \underset{\mathbf{T}^\top\mathbf{T}=\mathbf{I}}{\arg\min} ||\mathbf{Z}^{(i)}_{DGA+}\mathbf{T} - \mathbf{Z}^{(0)}_{DGA+}||_F \quad (12)$$

for every SR, where $\mathbf{Z}^{(i)}_{DGA+}$ is the embedding matrix of the SR $i$, and $\mathbf{Z}^{(0)}_{DGA+}$ is the embedding matrix of a randomly chosen SR (which is the same for all projections). We then compute the intersection of stem and affix group nodes from all SRs $\mathcal{S}_\cap = \bigcap_i \mathcal{S}^{(i)}$ and $\mathcal{A}_\cap = \bigcap_i \mathcal{A}^{(i)}$, where $\mathcal{S}^{(i)}$ and $\mathcal{A}^{(i)}$ are the stem and affix group sets of SR $i$, respectively. To probe whether differences between SRs are larger or smaller for affix embeddings as compared to stem embeddings, we define

$$\Delta(\mathcal{S}^{(i)}, \mathcal{S}^{(j)}) = \sum_{s\in\mathcal{S}_\cap} \frac{\cos(\hat{\mathbf{z}}^{(i)}_s, \hat{\mathbf{z}}^{(j)}_s)}{|\mathcal{S}_\cap|}, \quad (13)$$

i.e., the mean cosine similarity between projected embedding pairs $\hat{\mathbf{z}}^{(i)}_s$ and $\hat{\mathbf{z}}^{(j)}_s$ from two SRs $i$ and $j$ representing the same stem $s$ in the intersection set $\mathcal{S}_\cap$, with $\hat{\mathbf{z}}^{(i)}_s = \mathbf{z}^{(i)}_s\mathbf{R}^{(i)}$. $\Delta(\mathcal{A}^{(i)}, \mathcal{A}^{(j)})$ is defined analogously for affix groups.

The mean value for $\Delta(\mathcal{A}^{(i)}, \mathcal{A}^{(j)})$ ($0.723 \pm 0.102$) is lower than that for $\Delta(\mathcal{S}^{(i)}, \mathcal{S}^{(j)})$ ($0.760 \pm 0.087$), i.e., differences between affix group embeddings are more pronounced than between stem embeddings. Topically connected SRs are more similar to each other than SRs of different topic groups,




(a) $\Delta(\mathcal{S}^{(i)}, \mathcal{S}^{(j)})$ (b) $\Delta(\mathcal{A}^{(i)}, \mathcal{A}^{(j)})$

Figure 6: Comparison of embedding spaces across SRs. The plots show color-coded values of $\Delta(\mathcal{S}^{(i)}, \mathcal{S}^{(j)})$ and $\Delta(\mathcal{A}^{(i)}, \mathcal{A}^{(j)})$ for all pairs of SRs, respectively. The block-diagonal structure highlights the impact of topical relatedness on embedding similarities.

with the differences being larger in $\Delta(\mathcal{A}^{(i)}, \mathcal{A}^{(j)})$ than in $\Delta(\mathcal{S}^{(i)}, \mathcal{S}^{(j)})$ (see Figure 6).

These results can be related to Section 6.1: affix groups are very close to the stems they associate with in $\mathbf{Z}_{DGA+}$, i.e., if an affix group is used with stems of meaning $p$ in one SR and stems with meaning $q$ in the other SR, then the affix groups also have embeddings close to $p$ and $q$ in the two SRs. Most technical vocabulary, on the other hand, is specific to a SR and does not make it into $\mathcal{S}_{\cap}$.[8]

A qualitative analysis supports this hypothesis: affix groups with low cosine similarities between SRs associate with highly topical stems; e.g., the affix group `$ocracy` has a low cosine similarity of -0.189 between the SRs nba and pol, and it occurs with stems such as `kobe`, `jock` in nba but `left`, `wealth` in pol.

## 7 Related Work

Much recent computational research on **derivational morphology in NLP** has focused on two related problems: predicting the meaning of a derivative given its form, and predicting the form of a derivative given its meaning.

The first group of studies models the meaning of derivatives as a function of their morphological structure by training embeddings directly on text segmented into morphemes (Luong et al., 2013; Qiu et al., 2014) or by inferring morpheme embeddings from whole-word vector spaces, e.g., using the vector offset method (Lazaridou et al., 2013; Padó et al., 2016). Formally, given a derived form $f_d$, this line of research tries to find the meaning $m_d$ that maximizes $P(m_d|f_d)$.

The second group of studies models the form of derivatives as a function of their meaning. The meaning is represented by the base word and a semantic tag (Cotterell et al., 2017; Deutsch et al., 2018) or the sentential context (Vylomova et al., 2017). Formally, given a meaning $m_d$, these studies try to find the derived form $f_d$ of a word that maximizes $P(f_d|m_d)$.

Our study differs from these two approaches in that we model $P(W|f_d, m_d)$, i.e., we predict the overall likelihood of a derivative to exist. For future research, it would be interesting to apply derivational embeddings in studies of the second type by using them as pretrained input.

**Neural link prediction** is the task of inferring the existence of unknown connections between nodes in a graph. Advances in deep learning have prompted various neural models for link prediction that learn distributed node representations (Tang et al., 2015; Grover and Leskovec, 2016). Kipf and Welling (2016, 2017) proposed a convolutional graph auto-encoder that allows to include feature vectors for each node. The model was adapted to bipartite graphs by van den Berg et al. (2018).

Previous studies on neural link prediction for bipartite graphs have shown that the embeddings of the two node sets should ideally form separated clusters (Gao et al., 2018). Our work demonstrates that relations transcending the two-mode graph structure can lead to a trade-off between clustering and dispersion in embedding space.

## 8 Conclusion

We have introduced a derivational graph auto-encoder (DGA) that combines syntactic and semantic information with associative information from the mental lexicon to predict morphological well-formedness (MWF), a task that has not been addressed before. The model achieves good results and performs on par with a character-based LSTM at a fraction of the number of trainable parameters (less than 10%). Furthermore, the model learns embeddings capturing information about the compatibility of affixes and stems in derivation.

**Acknowledgements.** Valentin Hofmann was funded by the Arts and Humanities Research Council and the German Academic Scholarship Foundation. This research was also supported by the European Research Council (Grant No. 740516). We thank the reviewers for their helpful and very constructive comments.

[8] One SR standing out in Figure 6 is lol, a multiplayer online video game, in which many common stems such as `fame` and `range` have highly idiosyncratic meanings.

## References


Paolo Acquaviva. 2016. Morphological semantics. In Andrew Hippisley and Gregory Stump, editors, *The Cambridge handbook of morphology*, pages 117–148. Cambridge University Press, Cambridge.

Lada A. Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social Networks*, 25:211–230.

Maria Alegre and Peter Gordon. 1999. Rule-based versus associative processes in derivational morphology. *Brain and Language*, 68(1-2):347–354.

Margaret R. Allen. 1979. *Morphological investigations*. University of Connecticut, Mansfield, CT.

Frank Anshen and Mark Aronoff. 1999. Using dictionaries to study the mental lexicon. *Brain and Language*, 68:16–26.

Mark Aronoff. 1976. *Word formation in generative grammar*. MIT Press, Cambridge, MA.

Laurie Bauer. 2001. *Morphological productivity*. Cambridge University Press, Cambridge, UK.

Laurie Bauer. 2019. *Rethinking morphology*. Edinburgh University Press, Edinburgh, UK.

Joan Bybee. 1995. Regular morphology and the lexicon. *Language and Cognitive Processes*, 10(425-455).

Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In *International Conference on Knowledge Discovery and Data Mining (KDD) 23*.

Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov, and David Yarowsky. 2017. Paradigm completion for derivational morphology. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2017*.

David Crystal. 1997. *The Cambridge encyclopedia of the English language*. Cambridge University Press, Cambridge, UK.

Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning (ICML) 33*.

Marco del Tredici and Raquel Fernández. 2018. The road to success: Assessing the fate of linguistic innovations in online communities. In *International Conference on Computational Linguistics (COLING) 27*.

Daniel Deutsch, John Hewitt, and Dan Roth. 2018. A distributional and orthographic aggregation model for english derivational morphology. In *Annual Meeting of the Association for Computational Linguistics (ACL) 56*.

Nigel Fabb. 1988. English suffixation is constrained only by selectional restrictions. *Natural Language & Linguistic Theory*, 6(4):527–539.

Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. BiNE: Bipartite network embedding. In *International Conference on Research and Development in Information Retrieval (SIGIR) 41*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML) 34*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *International Conference on Knowledge Discovery and Data Mining (KDD) 22*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Annual Meeting of the Association for Computational Linguistics (ACL) 49*.

Martin Haspelmath and Andrea D. Sims. 2010. *Understanding morphology*. Routledge, New York, NY.

Jennifer Hay. 2001. Lexical frequency in morphology: Is everything relative? *Linguistics*, 39(6):1041–1070.

Jennifer Hay and Ingo Plag. 2004. What constrains possible suffix combinations? on the interaction of grammatical and processing restrictions in derivational morphology. *Natural Language & Linguistic Theory*, 22(3):565–596.

Valentin Hofmann, Janet B. Pierrehumbert, and Hinrich Schütze. 2020. Predicting the growth of morphological families from social and linguistic factors. In *Annual Meeting of the Association for Computational Linguistics (ACL) 58*.

Diederik P. Kingma and Jimmy L. Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR) 3*.

Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. In *NIPS Bayesian Deep Learning Workshop*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR) 5*.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Annual Meeting of the Association for Computational Linguistics (ACL) 51*.

David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *ACM Conference on Information and Knowledge Management (CIKM) 12*.

Pierre Lison and Andrey Kutuzov. 2017. Redefining context windows for word embedding models: An experimental study. In *arXiv 1704.05781*.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Conference on Computational Natural Language Learning (CoNLL) 17*.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL) 45*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML) 27*.

Sebastian Padó, Aurélie Herbelot, Max Kisselew, and Jan Šnajder. 2016. Predictability of distributional semantics in derivational word formation. In *International Conference on Computational Linguistics (COLING) 26*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*.

Janet Pierrehumbert. 2012. The dynamic lexicon. In Abigail Cohn, Cécile Fougeron, and Marie Huffman, editors, *The Oxford handbook of laboratory phonology*, pages 173–183. Oxford University Press, Oxford.

Janet Pierrehumbert and Ramon Granell. 2018. On hapax legomena and morphological productivity. In *Workshop on Computational Research in Phonetics, Phonology, and Morphology (SIGMORPHON) 15*.

Ingo Plag. 2003. *Word-formation in English*. Cambridge University Press, Cambridge, UK.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *International Conference on Computational Linguistics (COLING) 25*.

Peter H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 36(1).

Ingrid Sonnenstuhl and Axel Huth. 2002. Processing and representation of german -n plurals: A dual mechanism approach. *Brain and Language*, 81(1-3):276–290.

S. Sreelekha and Pushpak Bhattacharyya. 2018. Morphology injection for English-Malayalam statistical machine translation. In *International Conference on Language Resources and Evaluation (LREC) 11*.

Gregory Stump. 2017. Rule conflation in an inferential-realizational theory of morphotactics. *Acta Linguistica Academica*, 64(1):79–124.

Gregory Stump. 2019. Some sources of apparent gaps in derivational paradigms. *Morphology*, 29(2):271–292.

Wanhua Su, Yan Yuan, and Mu Zhu. 2015. A relationship between the average precision and the area under the ROC curve. In *International Conference on the Theory of Information Retrieval (ICTIR) 2015*.

Monica Tamariz. 2008. Exploring systematicity between phonological and context-cooccurrence representations of the mental lexicon. *The Mental Lexicon*, 3(2):259–278.

Chenhao Tan and Lillian Lee. 2015. All who wander: On the prevalence and characteristics of multi-community engagement. In *International Conference on World Wide Web (WWW) 24*.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *International Conference on World Wide Web (WWW) 24*.

Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2018. Graph convolutional matrix completion. In *KDD 2018 Deep Learning Day*.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579 2605.

Ekaterina Vylomova, Ryan Cotterell, Timothy Baldwin, and Trevor Cohn. 2017. Context-aware prediction of derivational word-forms. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL) 15*.

# A Appendices

## A.1 Data Preprocessing

We filter the Reddit posts for known bots and spammers (Tan and Lee, 2015). We remove abbreviations, strings containing numbers, references to users and SRs, and both full and shortened hyperlinks. We convert British English spelling variants to American English and lemmatize all words. We follow Han and Baldwin (2011) in reducing repetitions of more than three letters (`niiiiice`) to three letters. Except for excluding stopwords, we do not employ a frequency threshold.

| | sports | | | | | | entertainment | | | | | | knowledge | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cfb | | nba | | nfl | | gam | | lol | | mov | | pol | | sci | | tec | | $\mu \pm \sigma$ | |
| Model | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC | AP | AUC |
| JS | .632 | .593 | .617 | .582 | .626 | .588 | .619 | .588 | .609 | .584 | .622 | .589 | .614 | .591 | .649 | .617 | .638 | .608 | .625±.012 | .593±.011 |
| AA | .603 | .556 | .599 | .556 | .602 | .553 | .605 | .561 | .589 | .553 | .596 | .552 | .592 | .556 | .606 | .558 | .606 | .562 | .600±.006 | .556±.003 |
| CN | .600 | .553 | .596 | .553 | .598 | .550 | .602 | .558 | .585 | .550 | .592 | .548 | .588 | .552 | .601 | .554 | .603 | .558 | .596±.006 | .553±.003 |
| PA | .537 | .517 | .543 | .527 | .542 | .522 | .559 | .545 | .545 | .534 | .533 | .519 | .541 | .534 | .513 | .503 | .537 | .526 | .539±.011 | .525±.011 |

Table 5: Performance on MWF prediction. The table shows AP and AUC of the models for the nine Subreddits as well as averaged scores. Grey highlighting illustrates the best score in a column, light grey the second-best.

### A.2 Morphological Segmentation

We start by defining a set of potential stems $O^{(i)}$ for each Subreddit $i$. A word $w$ is given the status of a potential stem and added to $O^{(i)}$ if it consists of at least 4 characters and has a frequency count of at least 100 in the Subreddit.

Then, to determine the stem of a specific word $w$, we employ an iterative algorithm. Let $V^{(i)}$ be the vocabulary of the Subreddit, i.e., all words occurring in it. Define the set $B_1$ of $w$ as the bases in $V^{(i)}$ that remain when one affix is removed, and that have a higher frequency count than $w$ in the Subreddit. For example, `reaction` can be segmented as `re$action` and `react$ion`, so $B_1(\texttt{reaction}) = \{\texttt{action}, \texttt{react}\}$ (assuming `action` and `react` both occur in the Subreddit and are more frequent than `reaction`). We then iteratively create $B_{i+1}(w) = \bigcup_{b \in B_i(w)} B_1(b)$. Let further $B_0(w) = \{w\}$. We define $S(w) = O^{(i)} \cap B_m(w)$ with $m = \max\{k | O^{(i)} \cap B_k(w) \neq \emptyset\}$ as the set of stems of $w$. If $|S(w)| > 1$ (which is rarely the case in practice), the element with the lowest number of suffixes is chosen.

The algorithm is sensitive to most morpho-orthographic rules of English (Plag, 2003): when `$ness` is removed from `happi$ness`, e.g., the result is `happy`, not `happi`.

### A.3 Message Passing Rule

Let $\hat{\mathbf{B}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ be the adjacency matrix of the DG $\mathcal{B}$ with added self-loops, i.e., $\hat{B}_{ii} = 1$ and $\hat{\mathbf{D}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ the degree matrix of $\hat{\mathbf{B}}$ with $\hat{D}_{ii} = \sum_j \hat{B}_{ij}$. The matrix form of the message passing step can be expressed as

$$\mathbf{M}^{(l)} = \hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{B}} \hat{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X}^{(l-1)} \mathbf{W}^{(l)}, \tag{14}$$

where $\mathbf{W}^{(l)}$ is the trainable weight matrix of layer $l$, and $\mathbf{X}^{(l-1)} \in \mathbb{R}^{|\mathcal{V}| \times |n|}$ is the matrix containing the node feature vectors from layer $l-1$ (Kipf and Welling, 2016, 2017). The activation step then is

$$\mathbf{X}^{(l)} = \mathrm{ReLU}\left(\mathbf{M}^{(l)}\right). \tag{15}$$

### A.4 Feature-based Link Prediction

Besides Jaccard similarity, we implement three other feature-based link prediction methods.

**Adamic-Adar.** The Adamic-Adar (AA) index (Adamic and Adar, 2003) has to take the bipartite structure of DGs into account. Using the modified definition of common neighbors as with $\zeta_{JS}$, we calculate it as

$$\zeta_{AA}(s,a) = \sum_{\substack{n \in \Gamma_\cap(s,a) \\ n \in \Gamma_\cap(a,s)}} \frac{1}{d(n)}. \tag{16}$$

**Common Neighbors.** The score of an edge $(s,a)$ is calculated as the cardinality of the set of common neighbors (CN) of $s$ and $a$. Similarly to $\zeta_{JS}$ and $\zeta_{AA}$, we calculate the CN score as

$$\zeta_{CN}(s,a) = |\Gamma_\cap(s,a)| + |\Gamma_\cap(a,s)|. \tag{17}$$

**Preferential Attachment.** For preferential attachment (PA), the score of an edge $(s,a)$ is the product of the two node degrees,

$$\zeta_{PA}(s,a) = d(s) \cdot d(a). \tag{18}$$

The training regime is identical to Jaccard similarity. AA outperforms PA and CN but is consistently beaten by JS (see Table 5).