Morphological Smoothing and Extrapolation of Word Embeddings

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
CIS
LMU Munich, Germany
inquiries@cis.lmu.org

Jason Eisner
Department of Computer Science
Johns Hopkins University, USA
jason@cs.jhu.edu

Abstract

Languages with rich inflectional morphology exhibit lexical data sparsity, since the word used to express a given concept will vary with the syntactic context. For instance, each count noun in Czech has 12 forms (where English uses only singular and plural). Even in large corpora, we are unlikely to observe all inflections of a given lemma. This reduces the vocabulary coverage of methods that induce continuous representations for words from distributional corpus information. We solve this problem by exploiting existing morphological resources that can enumerate a word's component morphemes. We present a latent-variable Gaussian graphical model that allows us to extrapolate continuous representations for words not observed in the training corpus, as well as smoothing the representations provided for the observed words. The latent variables represent embeddings of morphemes, which combine to create embeddings of words. Over several languages and training sizes, our model improves the embeddings for words, when evaluated on an analogy task, skip-gram predictive accuracy, and word similarity.

1 Introduction

Representations of words as high-dimensional real vectors have been shown to benefit a wide variety of NLP tasks. Because of this demonstrated utility, many aspects of vector representations have been explored recently in the literature. One of the most interesting discoveries is that these representations capture meaningful morpho-syntactic and semantic properties through very simple linear relations: in a semantic vector space, we observe that

$$v_{talked} - v_{talk} \approx v_{drank} - v_{drink}. \quad (1)$$

That this equation approximately holds across many morphologically related 4-tuples indicates that the learned embeddings capture a feature of English morphology—adding the past tense feature roughly corresponds to adding a certain vector. Moreover, manipulating this equation yields what we will call the **vector offset method** (Mikolov et al., 2013c) for approximating other vectors. For instance, if we only know the vectors for the Spanish words *comieron* (*ate*), *comemos* (*eat*) and *bebieron* (*drank*), we can produce an approximation of the vector for *bebemos* (*drink*), as shown in Figure 1.

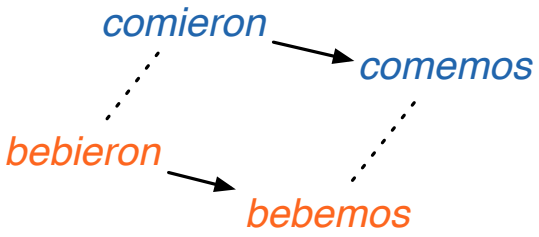

Figure 1: A visual depiction of the vector offset method for morpho-syntactic analogies in $\mathbb{R}^2$. We expect *bebieron* and *bebemos* to have the same relation (vector offset shown as solid vector) as *comieron* and *comemos*.

Many languages exhibit much richer morphology than English. While English nouns commonly take two forms – singular and plural—Czech nouns take 12 and Turkish nouns take over 30. This increase in word forms per lemma creates considerable data sparsity. Fortunately, for many languages there exist large morphological lexicons, or better yet, morphological tools that can analyze *any* word form—meaning that we have analyses (usually accurate) for forms that were unobserved or rare in our training corpus.

Our proposed method runs as a fast post-processor (taking under a minute to process 100-dimensional embeddings of a million observed word types) on the output of any existing tool that constructs word embeddings, such as WORD2VEC.

	Indicative		Subjunctive	
	Sg	Pl	Sg	Pl
1	bebo	bebemos	beba	bebamos
2	bebes	bebéis	bebas	bebáis
3	bebe	beben	beba	beban

Table 1: The paradigm of the Spanish verb BEBER (to drink). The paradigm actually consists of > 40 word forms; only the present tense portion is shown here.

In this output, some embeddings are noisy or missing, due to sparse training data. We correct these problems by using a Gaussian graphical model that jointly models the embeddings of morphologically related words. Inference under this model can smooth the noisy embeddings that were observed in the WORD2VEC output. In the limiting case of a word for which no embedding was observed (equivalent to infinite noise), inference can extrapolate one based on the observed embeddings of related words—a kind of global version of the vector offset method. The *structure* of our graphical model is defined using morphological lexicons, which supply analyses for each word form.

We conduct a comprehensive study of our ability to modify and generate vectors across five languages. Our model also dramatically improves performance on the morphological analogy task in many cases: e.g., accuracy at selecting the nominative plural forms of Czech nouns is 89%, ten times better than the standard analogy approach.

2 Background: Inflectional Morphology

Many languages require every verb token to be inflected for certain properties, such as person, number, tense, and mood. A verbal **paradigm** such as Table 1 lists all the inflected forms of a given verb. We may refer to this verb in the abstract by its **lemma**, BEBER—but when using it in a sentence, we must instead select from its paradigm the word type, such as *bebéis*, that expresses the contextually appropriate properties. Noun tokens in a language may similarly be required to be inflected for properties such as case, gender, and number.

A content word is chosen by specifying a lemma (which selects a particular paradigm) together with some inflectional attributes (which select a particular slot within that paradigm). For example, [Lemma=EAT, Person=3, Number=SINGULAR, Tense=PRESENT] is a bundle of attribute-value pairs that would be jointly expressed in English by the word form *eats* (Sylak-Glassman et al., 2015).

The regularities observed by Mikolov et al. (2013c) hold between words with similar attribute-value pairs. In Spanish, the word *beben* "they drink" (Table 1) can be analyzed as expressing the bundle [Lemma=BEBER, Person=3, Number=PLURAL, Tense=PRESENT]. Its vector similarity to *bebemos* "we drink" is due to the fact that both word forms have the same lemma BEBER. Likewise, the vector similarity of *beben* to *comieron* "they ate" is due to the conceptual similarity of their lemmas, BEBER "drink" and COMER "eat". Conversely, that *beben* is similar to *preguntan* "they ask" is caused by shared inflectional attributes [Person=3, Number=PLURAL, Tense=PRESENT]. Under cosine similarity, the most similar words are often related on both axes at once: e.g., one of the word forms closest to *beben* typically is *comen* "they eat".

3 Approach

Following this intuition, we fit a directed Gaussian graphical model (GGM) that simultaneously considers (i) each word's embedding (obtained from an embedding model like WORD2VEC) and (ii) its morphological analysis (obtained from a lexical resource). We then use this model to smooth the provided embeddings, and to generate embeddings for unseen inflections. For a lemma covered by the resource, the GGM can produce embeddings for all its forms (if at least one of these forms has a known embedding); this can be extended to words not covered using a guesser like MORFESSOR (Creutz and Lagus, 2007) or CHIPMUNK (Cotterell et al., 2015a).

A major difference of our approach from related techniques is that our model uses existing morphological resources (e.g., morphological lexicons or finite-state analyzers) rather than semantic resources (e.g., WordNet (Miller et al., 1990) and PPDB (Ganitkevitch et al., 2013)). The former tend to be larger: we often can analyze more words than we have semantic representations for.

It would be possible to *integrate* our GGM into the training procedure for a word embedding system, making that system sensitive to morphological attributes. However, the *postprocessing* approach in our present paper lets us use any existing word embedding system as a black box. It is simple to implement, and turns out to get excellent results, which will presumably improve further as

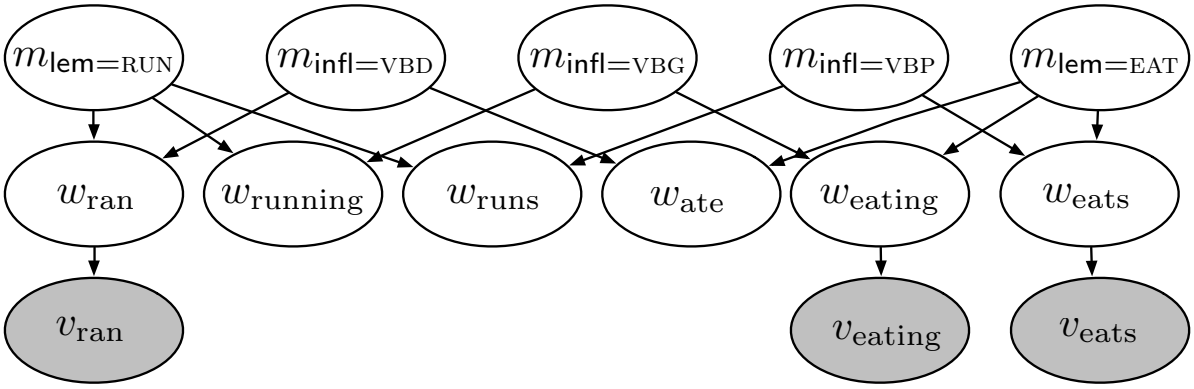

Figure 2: A depiction of our directed Gaussian graphical model (GGM) for the English verbal paradigm. Each variable represents a vector in $\mathbb{R}^n$; thus, this is not the traditional presentation of a GGM in which each node would be a *single real-valued random variable*, but each node represents a *real-valued random vector*. The shaded nodes v_i at the bottom are observed word embeddings. The nodes w_i at the middle layer are smoothed or extrapolated word embeddings. The nodes m_k at the top are latent embeddings of morphemes.

better black boxes become available.

4 A Generative Model

Figure 2 draws our GGM's structure as a Bayes net. In this paper, we loosely use the term "morpheme" to refer to an attribute-value pair (possibly of the form Lemma=...). Let $\mathcal{M}$ be the set of all morphemes. In our model, each morpheme $k \in \mathcal{M}$ has its *own* latent embedding $m_k \in \mathbb{R}^n$. These random variables are shown as the top layer of Figure 2. We impose an IID spherical Gaussian prior on them (similar to L_2 regularization with strength $\lambda > 0$):

$$m_k \sim \mathcal{N}(\mathbf{0}, \lambda^{-1} I), \ \forall k \tag{2}$$

Let $\mathcal{L}$ be the lexicon of all word types that appear in our lexical resource. (The noun and verb senses of *bat* are separate entries in $\mathcal{L}$.) In our model, each word $i \in \mathcal{L}$ has a latent embedding $w_i \in \mathbb{R}^n$. These random variables are shown as the middle layer of Figure 2. We assume that each w_i is simply a *sum* of the m_k for its component morphemes $M_i \subseteq \mathcal{M}$ (shown in Figure 2 as w_i's parents), *plus* a Gaussian perturbation:

$$w_i \sim \mathcal{N}(\sum_{k \in M_i} m_k, \Sigma_i), \ \forall i \tag{3}$$

This perturbation models idiosyncratic usage of word i that is not predictable from its morphemes. The covariance matrix Σ_i is shared for all words i with the same coarse POS (e.g., VERB).

Our system's *output* will be a guess of all of the w_i. Our system's *input* consists of noisy estimates v_i for some of the w_i, as provided by a black-box word embedding system run on some large corpus $\mathcal{C}$. (Current systems estimate the same vector for both senses of *bat*.) These observed random variables are shown as the bottom layer of Figure 2. We assume that the black-box system would have recovered the "true" w_i if given enough data, but instead it gives a noisy small-sample estimate

$$v_i \sim \mathcal{N}(w_i, \tfrac{1}{n_i}\Sigma_i'), \ \forall i \tag{4}$$

where n_i is the count of word i in training corpus $\mathcal{C}$.

This formula is inspired by the central limit theorem, which guarantees that v_i's distribution would approach (4) (as $n_i \to \infty$) if it were estimated by averaging a set of n_i noisy vectors drawn IID from *any* distribution with mean w_i (the truth) and covariance matrix Σ_i'. A system like WORD2VEC does not precisely do that, but it does choose v_i by aggregating (if not averaging) the influences from the contexts of the n_i tokens.

The parameters $\lambda, \Sigma_i, \Sigma_i'$ now have likelihood

$$p(\boldsymbol{v}) = \int p(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{m}) \, d\boldsymbol{w} \, d\boldsymbol{m}, \text{ where} \tag{5}$$

$$p(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{m}) = \prod_{k \in \mathcal{M}} p(m_k) \cdot \prod_{i \in \mathcal{L}} p(w_i \,|\, m_k : k \in M_i) \cdot p(v_i \,|\, w_i) \tag{6}$$

Here $\boldsymbol{m} = \{m_k : k \in \mathcal{M}\}$ represents the collection of all latent morpheme embeddings, and similarly $\boldsymbol{w} = \{w_i : i \in \mathcal{L}\}$ and $\boldsymbol{v} = \{v_i : i \in \mathcal{L}\}$. We take $p(v_i \mid w_i) = 1$ if no observation v_i exists.

How does the model behave qualitatively? If $\hat{w}_i$ is the MAP estimate of w_i, then $\hat{w}_i \to v_i$ as $n_i \to \infty$, but $\hat{w}_i \to \sum_{k \in M_i} m_k$ as $n_i \to 0$. This

is because (3) and (4) are in tension; when n_i is small, (4) is weaker and we get more smoothing. The morpheme embeddings m_k are largely determined from the observed embeddings v_i of the frequent words (since m_k aims via (2)–(3) to explain w_i, which $\approx v_i$ when i is frequent). That determines the compositional embedding $\sum_{k \in M_i} m_k$ toward which the w_i of a rarer word is smoothed (away from v_i). If v_i is not observed or if $n_i = 0$, then $\hat{w}_i = \sum_{k \in M_i} m_k$ exactly.

5 Inference

Suppose first that the model parameters are known, and we want to reconstruct the latent vectors w_i. Because the joint density $p(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{m})$ in (6) is a product of (sometimes degenerate) Gaussian densities, it is itself a highly multivariate Gaussian density over all elements of all vectors.[1] Thus, the posterior marginal distribution of each w_i is Gaussian as well. A good deal is known about how to exactly compute these marginal distributions of a Gaussian graphical model (e.g., by matrix inversion) or at least their means (e.g., by belief propagation) (Koller and Friedman, 2009).

For this paper, we adopt a simpler method—MAP estimation of all latent vectors. That is, we seek the $\boldsymbol{w}, \boldsymbol{m}$ that jointly maximize (6). This is equivalent to minimizing

$$\sum_k \lambda ||m_k||_2^2 + \sum_i ||w_i - \sum_{k \in M_i} m_k||_{\Sigma_i}^2 + \sum_i ||v_i - w_i||_{\Sigma_i'/n_i}^2, \quad (7)$$

which is a simple convex optimization problem.[2] We apply block coordinate descent until numerical convergence, in turn optimizing each vector m_k or w_i with all other vectors held fixed. This finds the global minimum (convex objective) and is extremely fast even when we have over a hundred million real variables. Specifically, we update

$$m_k \leftarrow \big(\lambda I + \sum_{i \in W_k} \mathfrak{T}_i\big)^{-1} \sum_{i \in W_k} \mathfrak{T}_i (w_i - \sum_{j \in M_i, j \neq k} m_j),$$

where $\mathfrak{T} \stackrel{\text{def}}{=} \Sigma^{-1}$ is the inverse covariance matrix and $W_k \stackrel{\text{def}}{=} \{i : k \in M_i\}$. This updates m_k so the partial derivatives of (7) with respect to the components of m_k are 0. In effect, this updates m_k to a *weighted average* of several vectors. Morpheme k participates in words $i \in W_k$, so its vector m_k is updated to the average of the contributions $(w_i - \sum_{j \in M_i, j \neq k} m_j)$ that m_k would *ideally* make to the embeddings w_i of those words. The contribution of w_i is "weighted" by the inverse covariance matrix $\mathfrak{T}_i$. Because of prior (2), $\mathbf{0}$ is also included in the average, "weighted" by λI.

Similarly, the update rule for w_i is

$$w_i \leftarrow (n_i \mathfrak{T}_i' + \mathfrak{T}_i)^{-1} \big(n_i \mathfrak{T}_i' v_i + \mathfrak{T}_i \sum_{k \in M_i} m_k\big),$$

which can similarly be regarded as a weighted average of the observed and compositional representations.[3] See Appendix C for the derivations.

6 Parameter Learning

We wish to optimize the model parameters $\lambda, \Sigma_i, \Sigma_i'$ by empirical Bayes. That is, we do not have a prior on these parameters, but simply do maximum likelihood estimation. A standard approach is the Expectation-Maximization or EM algorithm (Dempster et al., 1977) to *locally* maximize the likelihood. This alternates between reconstructing the latent vectors given the parameters (E step) and optimizing the parameters given the latent vectors (M step). In this paper, we use the Viterbi approximation to the E step, that is, MAP inference as described in section 5. Thus, our overall method is Viterbi EM.

As all conditional probabilities in the model are Gaussian, the M step has closed form. MLE estimation of a covariance matrix is a standard result—in our setting the update to Σ_i takes the form:

$$\Sigma_c \leftarrow \frac{1}{N_c} \sum_{i : c \in C(i)} (w_i - \sum_{k \in M_i} m_k)(w_i - \sum_{k \in M_i} m_k)^T,$$

where $C(i)$ are i's POS tags, $N_c = |\{i | c \in C(i)\}|$ and Σ_c is the matrix for the c^{th} POS tag (the matrices are tied by POS). In this paper we simply fix $\Sigma_i' = I$ rather than fitting it.[4] Also, we tune the hyperparameter λ on a development set, using grid search over the values $\{0.1, 0.5, 1.0\}$.

[1] Its inverse covariance matrix is highly sparse: its pattern of non-zeros is related to the graph structure of Figure 2. (Since the graphical model in Figure 2 is *directed*, the inverse covariance matrix has a sparse Cholesky decomposition that is even more directly related to the graph structure.)

[2] By definition, $||\boldsymbol{x}||_A^2 \stackrel{\text{def}}{=} \boldsymbol{x}^T A \boldsymbol{x}$.

[3] If v_i is not observed, take $n_i = 0$. In fact it is not necessary to represent this w_i during optimization. Simply omit i from all W_k. *After* convergence, set $w_i \leftarrow \sum_{k \in M_i} m_k$.

[4] Note that it is not necessary to define it as $\lambda' I$, introducing a new scale parameter λ', since doubling λ' would have the same effect on the MAP update rules as halving λ and Σ_i.

Viterbi EM can be regarded as block coordinate descent on the negative log-likelihood function, with E and M steps both improving this common objective along different variables. We update the parameters (M step above) after each 10 passes of updating the latent vectors (section 5's E step).

7 Related Work

Our postprocessing strategy is inspired by Faruqui et al. (2015), who designed a retrofitting procedure to modify pre-trained vectors such that their relations match those found in semantic lexicons. We focus on morphological resources, rather than semantic lexicons, and employ a generative model. More importantly, in addition to modifying vectors of observed words, our model can generate vectors for forms not observed in the training data.

Wieting et al. (2015) compute compositional embeddings of *phrases*, with their simplest method being additive (like ours) over the phrase's words. Their embeddings are tuned to fit observed phrase similarity scores from PPDB (Ganitkevitch et al., 2013), which allows them to smooth and extend PPDB just as we do to WORD2VEC output.

Using morphological resources to enhance embeddings at *training* time has been examined by numerous authors. Luong et al. (2013) used MORFESSOR (Creutz and Lagus, 2007), an unsupervised morphological induction algorithm, to segment the training corpus. They then trained a recursive neural network (Goller and Kuchler, 1996; Socher, 2014) to generate compositional word embeddings. Our model is much simpler and faster to train. Their evaluation was limited to English and focused on rare English words. dos Santos and Zadrozny (2014) introduced a neural tagging architecture (Collobert et al., 2011) with a character-level convolutional layer. Qiu et al. (2014) and Botha and Blunsom (2014) both use MORFESSOR segmentations to augment WORD2VEC and a log-bilinear (LBL) language model (Mnih and Hinton, 2007), respectively. Similar to us, they have an additive model of the semantics of morphemes, i.e., the embedding of the word form is the *sum* of the embeddings of its constituents. In contrast to us, however, both include the word form itself in the sum. Finally, Cotterell and Schütze (2015) jointly trained an LBL language model and a morphological tagger (Hajič, 2000) to encourage the embeddings to encode rich morphology. With the exception of (Cotterell and Schütze, 2015), all of the above methods use *unsupervised* methods to infuse word embeddings with morphology. Our approach is supervised in that we use a morphological lexicon, i.e., a manually built resource.

Our model is also related to other generative models of real vectors common in machine learning. The simplest of them is probabilistic principal component analysis (Roweis, 1998; Tipping and Bishop, 1999), a generative model of matrix factorization that explains a set of vectors via latent low-dimensional vectors. Probabilistic canonical correlation analysis similarly explains a set of *pairs* of vectors (Bach and Jordan, 2005).

Figure 2 has the same topology as our graphical model in (Cotterell et al., 2015b). In that work, the random variables were *strings* rather than vectors. Morphemes were combined into words by concatenating strings rather than adding vectors, and then applying a stochastic edit process (modeling phonology) rather than adding Gaussian noise.

8 Experiments

We perform three experiments to test the ability of our model to improve on WORD2VEC. To reiterate, our approach does not generate or analyze a word's spelling. Rather, it uses an *existing* morphological analysis of a word's spelling (constructed manually or by a rule-based or statistical system) as a resource to improve its embedding.

In our first experiment, we attempt to identify a corpus word that expresses a given set of morphological attributes. In our second experiment, we attempt to use a word's embedding to predict the words that appear in its context, i.e., the skip-gram objective of Mikolov et al. (2013a). Our third example attempts to use word embeddings to predict human similarity judgments.

We experiment on 5 languages: Czech, English, German, Spanish and Turkish. For each language, our corpus data consists of the full Wikipedia text. Table 5 in Appendix A reports the number of types and tokens and their ratio. The lexicons we use are characterized in Table 6: MorfFlex CZ for Czech (Hajič and Hlaváčová, 2013), CELEX for English and German (Baayen et al., 1993) and lexicons for Spanish and Turkish that were scraped from Wiktionary by Sylak-Glassman et al. (2015).

Given a finite training corpus $\mathcal{C}$ and a lexicon $\mathcal{L}$,[5] we generate embeddings v_i for all word

[5] $\mathcal{L}$ is finite in our experiments. It could be infinite (though still incomplete) if a morphological guesser were used.

	Nom Sg	Nom Pl	Gen Sg	Gen Pl	Dat Sg	Dat Pl	Acc Sg	Acc Pl	Ins Sg	Ins Pl	Voc Sg	Voc Pl
GGM	48	89	81	43	30	33	36	87	33	48	49	89
Nom Sg		4.4	2.6	1.7	3.9	0.33	2.1	1.6	3.5	0.59	0	5.2
Nom Pl	4.6		1.7	3.4	1.6	2.2	2.3	10	1.9	7	0	0
Gen Sg	2	1.7		2.9	0.67	0.5	0.98	0.5	1.3	1.6	0	0.33
Gen Pl	1.8	4.9	4.8		4.7	2.5	3.3	7.2	3.1	0.92	1.7	3.7
Dat Sg	2	2	0	3.3		0	1.3	0.61	2.8	0	0	2
Dat Pl	3.8	4.8	2.4	8	0.86		0.33	7.2	1	2.7	0	5
Acc Sg	3.9	2.7	0.74	2.1	1.7	0		2.4	0.58	2.5	5	2.2
Acc Pl	1.3	15	1.2	6.2	2.6	2.6	2.2		2.8	2.7	0	0
Ins Sg	4	1.6	1.6	2.3	2.7	0.36	1.6	2.4		0.18	0.83	2.9
Ins Pl	2	6	3	2.5	1.7	4.3	3.6	7.7	0.69		0	4.6
Voc Sg	0	0	0	0	0	5	5	0	0.83	0		0
Voc Pl	4.7	0	0	3.5	1.3	5.5	2.2	0	2.1	3.7	0	

	1ps Ps	2ps Ps	3ps Ps	1pp Ps	2pp Ps	3pp Ps	1ps Pt	2ps Pt	3ps Pt	1pp Pt	2pp Pt	3pp Pt
GGM	8.8	4.5	30	14	10	49	9.4	8.1	41	9.5	0	43
1ps Ps		0	0.79	2.1	0	0.33	0.83	0	3.4	1	0	3
2ps Ps	0		14	0.5	5	0.091	0.83	0	0.17	0	0	0.28
3ps Ps	1.2	9.3		1.2	0	29	1.7	3.3	13	2.5	0	8.2
1pp Ps	0.24	0	1.9		0	2.7	2.1	0	2.6	0	0	2.2
2pp Ps	0	0	0	0		0	0	0	0	0	0	0
3pp Ps	1.6	0	30	4	0		0.24	0	8	4.4	0	14
1ps Pt	0.14	0.5	2	3.2	0	0.33		0	1.4	0.83	0	1.7
2ps Pt	0	0	0	0	0	0	0		0	0	0	0
3ps Pt	7.6	0	13	1.3	0	7.2	0.86	0		3.5	0	36
1pp Pt	1.7	0	3.3	0	0	2.8	1.7	0	3.4		0	2
2pp Pt	0	0	0	0	0	0	0	0	0	0		0
3pp Pt	7.3	1.3	7.2	3.4	0	13	1.4	0	37	1.8	0	

Table 2: The two tables show how the Gaussian graphical model (GGM) compares to various analogies on Czech nouns (left) and Spanish verbs (right).The numbers in each cell represent the accuracy (larger is better). The columns represent the inflection of the word i to be predicted. Our GGM model is the top row. The other rows subdivide the baseline analogy results according to the inflection of source word a. Abbreviations: in the Czech noun table (left), the first word indicates the case and the second the number, e.g., Dat Sg = Dative Singular. In the Spanish verb table (right), the first word is the person and number and the second the tense, e.g., 3pp Pt = 3rd-person plural past.

types $i \in \mathcal{C}$, using the GENSIM implementation (Řehůřek and Sojka, 2010) of the WORD2VEC hierarchical softmax skip-gram model (Mikolov et al., 2013a), with a context size of 5. We set the dimension n to 100 for all experiments.[6]

We then apply our GGM to generate smoothed embeddings w_i for all word types $i \in \mathcal{C} \cap \mathcal{L}$. (Recall that the noun and verb sense of *bats* are separate types in $\mathcal{L}$, even if conflated in $\mathcal{C}$, and get separate embeddings.) How do we handle other word types? For an out-of-vocabulary (OOV) test word $i \notin \mathcal{C}$, we will extrapolate $w_i \leftarrow \sum_{k \in M_i} m_k$ on demand, as the GGM predicts, provided $i \in \mathcal{L}$. If any of these morphemes m_k were themselves never seen in $\mathcal{C}$, we back off to the mode of the prior to take $m_k = \mathbf{0}$.[7] Our experiments also encounter out-of-lexicon (OOL) test words $i \notin \mathcal{L}$, for which we have no morphological analysis; here we take $w_i = v_i$ (unsmoothed) if $i \in \mathcal{C}$ and $w_i = \mathbf{0}$ otherwise.

8.1 Experiment 1: Extrapolation vs. Analogy

Our first set of experiments uses embeddings for word selection. Our prediction task is to identify the unique word $i \in \mathcal{C}$ that expresses the morphological attributes M_i. To do this, we predict a target embedding x, and choose the most similar unsmoothed word by cosine distance, $\hat{\imath} = \text{argmax}_{j \in \mathcal{C}}\, v_j \cdot x$. We are scored correct if $\hat{\imath} = i$. Our experimental design ensures that $i \notin \mathcal{L}$, since if it were, we could trivially find i simply by consulting $\mathcal{L}$. The task is to identify *missing* lexical entries, by exploiting the distributional properties in $\mathcal{C}$.[8] Given the input bundle M_i, our method predicts the embedding $x = \sum_{k \in M_i} m_k$, and so looks for a word $j \in \mathcal{C}$ whose unsmoothed embedding $v_j \approx x$. The GGM's role here is to predict that the bundle M_i will be realized by something like x.

The baseline method is the analogy method of equation (1). This predicts the embedding x via the vector-offset formula $v_a + (v_b - v_c)$, where $a, b, c \in \mathcal{C} \cap \mathcal{L}$ are three other words sharing i's coarse part of speech such that M_i can be expressed as $M_a + (M_b - M_c)$.[9] Specifically, the baseline chooses a, b, c uniformly at random from all possibilities. (This is not too inefficient: given a, at most one choice of (b, c) is possible.) Note that the baseline extrapolates from the unsmoothed embeddings of 3 other words, whereas the GGM considers *all* words in $\mathcal{C} \cap \mathcal{L}$ that share i's morphemes.

[6] An additional important hyperparameter is the number of epochs. The default value in the GENSIM package is 5, which is suitable for larger corpora. We use this value for Experiments 1 and 3. Experiment 2 involves training on smaller corpora and we found it necessary to set the number of epochs to 10.

[7] One could in principle learn "backoff morphemes." For instance, if *borogoves* is analyzed as [Lemma=OOV_NOUN, Num=PL], we might want $m_{\text{Lemma=OOV\_NOUN}} \neq \mathbf{0}$ to represent novel *nouns*.

[8] The argmax selection rule does not exploit the fact that the entry is missing: it is free to incorrectly return some $\hat{\imath} \in \mathcal{L}$.

[9] More formally, $\overline{M_i} = \overline{M_a} + (\overline{M_b} - \overline{M_c})$, if we define $\overline{M}$ by $(\overline{M})_k = \mathbb{I}(k \in M)$ for all morphemes $k \in \mathcal{M}$. This converts morpheme bundle M to a $\{0, 1\}$ indicator vector $\overline{M}$ over $\mathcal{M}$.

	analogy	GGM		analogy	GGM		analogy	GGM		analogy	GGM		analogy	GGM
Ins^P	.067	.343*	Acc^S	.131	.582*	VBD	.052	.202*	Masc^S Part	.178	.340*	Loc^S	.002	.036*
Gen^S	.059	.632*	Nom^S	.136	.571*	VBG	.06	.102*	Fem^P Part	.230	.286	Abl^S	.001	.019*
Gen^P	.078	.242*	Dat^P	.074	.447*	NN	.051	.080*	Fem^S Part	.242	.235	Dat^S	.001	.037*
Nom^P	.076	.764*	Acc^P	.075	.682*	VBN	.052	.218*	Adj^S	.201	.286*	Acc^S	.001	.023*
Nom^S	.066	.290*	Gen^P	.075	.666*	NNS	.052	.114*	Gerund^S	.186	.449*	Ins^S	.001	.004
Voc^P	.063	.789*	Nom^P	.064	.665*	VB	.056	.275*	Gerund^P	.172	.311*	Nom^S	.001	.023*
Czech nouns			German nouns			English nouns & verbs			Spanish nouns, verbs & adj's			Turkish nouns		

Table 3: Test results for Experiment 1. The rows indicate the inflection of the test word i to be predicted (superscript ^{P} indicates plural, superscript ^{S} singular). The columns indicate the prediction method. Each number is an average over 10 training-test splits. Improvements marked with a $\star$ are statistically significant ($p < 0.05$) under a paired permutation test over these 10 runs.

Experimental Setup: A lexical resource consists of pairs (word form i, analysis M_i). For each language, we take a random 80% of these pairs to serve as the training lexicon $\mathcal{L}$ that is seen by the GGM. The remaining pairs are used to construct our prediction problems (given M_i, predict i), with a random 10% each as dev and test examples. We compare our method against the baseline method on ten such random training-test splits. We are releasing all splits for future research.

For some dev and test examples, the baseline method has no choice of the triple a, b, c. Rather than score these examples as incorrect, our baseline results do not consider them at all (which inflates performance). For each remaining example, to reduce variance, the baseline method reports the average performance on up to 100 a, b, c triples sampled uniformly without replacement.

The automatically created analogy problems ($a, b, c \rightarrow i$) solved by the baseline are similar to those of Mikolov et al. (2013c). However, most previous analogy evaluation sets evaluate only on 4-tuples of frequent words (Nicolai et al., 2015), to escape the need for smoothing, while ours also include infrequent words. Previous evaluation sets also tend to be translations of the original English datasets—leaving them impoverished as they therefore *only* test morpho-syntactic properties found *in English*. E.g., the German analogy problems of Köper et al. (2015) do not explore the four cases and two numbers in the German adjectival system. Thus our baseline analogy results are useful as a more comprehensive study *of the vector offset method for randomly sampled words*.

Results: Overall results for 5 languages are shown in Table 3. Additional rows break down performance by the inflection of the target word i. (The inflections shown are the ones for which the *baseline* method is most accurate.)

For almost all target inflections, GGM is significantly better than the analogy baseline. An extreme case is the vocative plural in Czech, for which GGM predicts vectors better by more than 70%. In other cases, the margin is slimmer; but GGM loses only on predicting the Spanish feminine singular participle. For Czech, German, English and Spanish the results are clear—GGM yields better predictions. This is not surprising as our method incorporates information from *multiple* morphologically related forms.

More detailed results for two languages are given in Table 2. Here, each row constrains the source word a to have a certain inflectional tag; again we average over up to 100 analogies, now chosen under this constraint, and again we discard a test example i from the test set if no such analogy exists. The GGM row considers all test examples.

Past work on morphosyntactic analogies has generally constrained a to be the unmarked (lemma) form (Nicolai et al., 2015). However, we observe that it is easier to predict one word form from another starting from a form that is "closer" in morphological space. For instance, it is easier to predict Czech forms inflected in the genitive plural from forms in nominative plural, rather than the nominative singular. Likewise, it is easier to predict a singular form from another singular form rather than from a plural form. It also is easier to predict partially syncretic forms, i.e., two inflected forms that share the same orthographic string; e.g., in Czech the nominative plural and the accusative plural are identical for inanimate nouns.

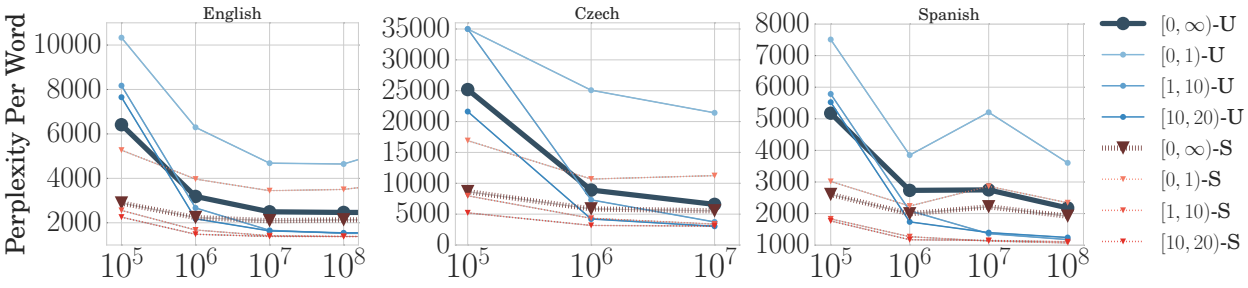

Figure 3: Results for the WORD2VEC skip-gram objective score (perplexity per predicted context word) on a held-out test corpus. The x-axis measures the size in tokens of the training corpus used to generate the model. We plot the held-out perplexity for the skip-gram model with Unsmoothed observed vectors $\boldsymbol{v}$ (solid ●) and Smoothed vectors $\boldsymbol{w}$ (barred ▼). The thickest, darkest curves show aggregate performance. The thinner, lighter versions show a breakdown according to whether the *predicting* word's frequency in the *smallest* training corpus falls in the range $[0, 1)$, $[1, 10)$, or $[10, 20)$ (from lightest to darkest and roughly from top to bottom). (These are the words whose representations we smooth; footnote 10 explains why we do not smooth the predicted context word.) We do not show $[20, \infty)$ since WORD2VEC randomly removes some tokens of high-frequency words ("subsampling"), similar in spirit to removing stop words. See Appendix B for more graphs.

8.2 Experiment 2: Held-Out Evaluation

We now evaluate the smoothed and extrapolated representations w_i. Fundamentally, we want to know if our approach improves the embeddings of the entire vocabulary, as if we had seen more evidence. But we cannot simply compare our smoothed vectors to "gold" vectors trained on much more data, since two different runs of WORD2VEC will produce incomparable embedding schemes. We must ask whether our embeddings improve results on a downstream task.

To avoid choosing a downstream task with a narrow application, we evaluate our embedding using the WORD2VEC skip-gram objective on *held-out* data—as one would evaluate a language model. If we believe that a better score on the WORD2VEC objective indicates generally more useful embeddings—which indeed we do as we optimize for it—then improving this score indicates that our smoothed vectors are superior. Concretely, the objective is

$$\sum_s \sum_t \sum_{j \in [t-5, t+5], j \neq t} \log_2 p_{\text{word2vec}}(T_{sj} \mid T_{st}), \quad (8)$$

where T_s is the s^{th} sentence in the *test* corpus, t indexes its tokens, and j indexes tokens near t. The probability model p_{word2vec} is defined in Eq. (3) of (Mikolov et al., 2013b). It relies on an embedding of the word form T_{st}.[10] Our baseline approach simply uses WORD2VEC's embeddings (or **0** for OOV words $T_{st} \notin \mathcal{C}$). Our GGM approach substitutes "better" embeddings when T_{st} appears in the lexicon $\mathcal{L}$ (if T_{st} is ambiguous, we use the mean w_i vector from all $i \in \mathcal{L}$ with spelling T_{st}).

Note that (8) is itself a kind of task of predicting words in context, resembling language modeling or a "cloze" task. Also, Taddy (2015) showed how to use this objective for document classification.

Experimental Setup: We evaluate GGM on the same 5 languages, but now hold out part of the corpus instead of part of the lexicon. We take the training corpus $\mathcal{C}$ to be the initial portion of Wikipedia of size 10^5, 10^6, 10^7 or 10^8. (We skip the 10^8 case for the smaller datasets: Czech and Turkish). The 10^7 tokens after that are the dev corpus; the next 10^7 tokens are the test corpus.

Results: We report results on three languages in Figure 3 and all languages in Appendix B. Smoothing from v_i to w_i helps a lot, reducing perplexity by up to 48% (Czech) with 10^5 training tokens and up to 10% (Spanish) even with 10^8 training tokens. This roughly halves the perplexity, which in the case of 10^5 training tokens, is equivalent to $8\times$ more training data. This is a clear win for lower-resource languages. We get larger gains from smoothing the rarer predicting words, but even words with frequency $\geq 10^{-4}$ benefit. (The exception is Turkish, where the large gains are confined to rare predicting words.) See Appendix B for more analysis.

[10] In the hierarchical softmax version, it also relies on a *separate* embedding for a variable-length bit-string encoding of the context word T_{sj}. Unfortunately, we do *not* currently know of a way to smooth these bit-string encodings (also found by WORD2VEC). However, it might be possible to directly incorporate morphology into the construction of the vocabulary tree that defines the bit-strings.

	English	German	Spanish
Forms / Lemma	1.8	6.3	8.1
Skip-Gram	**58.9**	36.2	37.8
GGM	**58.9**	**37.6**	**40.3**

Table 4: Word similarity results (correlations) using the WS-353 dataset in the three languages, in which it is available. Since all the words in WS-353 are lemmata, we report the average inflected form to lemma ratio for forms *appearing in* the datasets.

8.3 Experiment 3: Word Similarity

As a third and final experiment, we consider word similarity using the WS-353 data set (Finkelstein et al., 2001), translated into Spanish (Hassan and Mihalcea, 2009) and German (Leviant, 2016).[11] The datasets are composed of 353 pairs of words. Multiple native speakers were then asked to give an integral value between 1 and 10 indicating the similarity of that pair, and those values were then averaged. In each case, we train the GGM on the *whole* Wikipedia corpus for the language. Since in each language every word in the WS-353 set is in fact a lemma, we use the latent embedding our GGM learns in the experiment. In Spanish, for example, we use the learned latent morpheme embedding for the lemma BEBER (recall this takes information from every element in the paradigm, e.g., *bebemos* and *beben*), rather than the embedding for the infinitival form *beber*. In highly inflected languages we expect this to improve performance, because to get the embedding of a lemma, it leverages the distributional signal from *all* inflected forms of that lemma, not just a single one. Note that unlike previous retrofitting approaches, we do not introduce new semantic information into the model, but rather simply allow the model to better exploit the distributional properties already in the text, by considering words with related lemmata together. In essence, our approach embeds a lemma as the average of all words containing that lemma, *after* "correcting" those forms by subtracting off their other morphemes (e.g., inflectional affixes).

Results: As is standard in the literature, we report Spearman's correlation cofficient ρ between the averaged human scores and the cosine distance between the embeddings. We report results in Table 4. We additionally report the average number of forms per lemma. We find that we improve performance on the Spanish and German datasets over the original skip-gram vectors, but only equal the performance on English. This is not surprising as German and Spanish have roughly 3 and 4 times more forms per lemma than English. We speculate that cross-linguistically the GGM will improve word similarity scores more for languages with richer morphology.

[11] This dataset has yet to be translated into Czech or Turkish, nor are there any comparable resources in these languages.

9 Conclusion and Future Work

For morphologically rich languages, we generally will *not* observe, even in a large corpus, a high proportion of the word forms that exist in lexical resources. We have presented a Gaussian graphical model that exploits lexical relations documented in existing morphological resources to smooth vectors for observed words and extrapolate vectors for new words. We show that our method achieves large improvements over strong baselines for the tasks of morpho-syntactic analogies and predicting words in context. Future work will consider the role of derivational morphology in embeddings as well as noncompositional cases of inflectional morphology.

Acknowledgments

The first author was funded by a DAAD Long-Term Research Grant. This work was also partially supported by Deutsche Forschungsgemeinschaft (grat SCHU-2246/2-2 WordGraph) and by the U.S. National Science Foundation under Grant No. 1423276.

References

R Harald Baayen, Richard Piepenbrock, and Rijn van H. 1993. The CELEX lexical data base on CD-ROM.

Francis R Bach and Michael I Jordan. 2005. A probabilistic interpretation of canonical correlation analysis. Technical report, UC Berkeley.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *ICML*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word embeddings. In *NAACL*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015a. Labeled morphological segmentation with semi-Markov models. In *CoNLL*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015b. Modeling word forms using latent underlying morphs and phonology. *TACL*, 3.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):3.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, pages 1–38.

Cıcero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*. ACM.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*.

Jan Hajič and Jaroslava Hlaváčová. 2013. MorfFlex CZ. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *NAACL*.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *EMNLP*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual reliability and semantic structure of continuous word spaces. In *IWCS*.

Ira Leviant. 2016. *Separated by an Un-common Language: Towards Judgment Language Informed Vector Space Modeling*. Ph.D. thesis, Technion.

Minh-Thang Luong, Richard Socher, and C Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *NAACL*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Morpho-syntactic regularities in continuous word representations: A multilingual study. In *Proceedings of Workshop on Vector Space Modeling for NLP*, pages 129–134.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Liu Tie-Yan. 2014. Co-learning of word representations and morpheme representations. In *COLING*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Sam Roweis. 1998. EM algorithms for PCA and SPCA. In *NIPS*.

Richard Socher. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Ph.D. thesis, Stanford University.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *ACL*.

Matt Taddy. 2015. Document classification by inversion of distributed language representations. In *ACL*.

Michael E Tipping and Christopher M Bishop. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society B*, 61(3):611–622.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*.

Appendices

A Morphological Lexicons and Training Corpora

Here we provide additional information about the training corpora for the skip-gram model and the morphological lexicons used by our Gaussian graphical model.

	Tokens	Types	TTR
Czech	83,048,682	1,663,361	0.02
English	1,778,938,441	7,403,109	0.004
German	609,406,708	8,582,032	0.0141
Spanish	396,443,513	6,719,014	0.0169
Turkish	58,746,962	2,272,946	0.0387

Table 5: The number of types and tokens and their ratio (TTR) in each Wikipedia corpus.

	Lexicon	Types	OOV
Czech	MorfFlex	25,226,946	97.9
English	CELEX	79,208	2.3
German	CELEX	365,530	43.9
Spanish	Wiktionary	668,681	66.1
Turkish	Wiktionary	118,786	32.2

Table 6: Sizes of the various morphological lexicons and their origin. We note that our method is compatible with any morphologically annotated lexicon as well as finite-state analyzers that have the capacity to analyze an unbounded number of words. We also report OOV, the percentage of the types in the morphological lexicon that are *not* attested in the Wikipedia corpus.

B Additional Results

We include additional results for the experiment in section 8.2. In addition to showing results for all our test languages in Figure 4, we also show a different breakdown of the results in Figure 5.

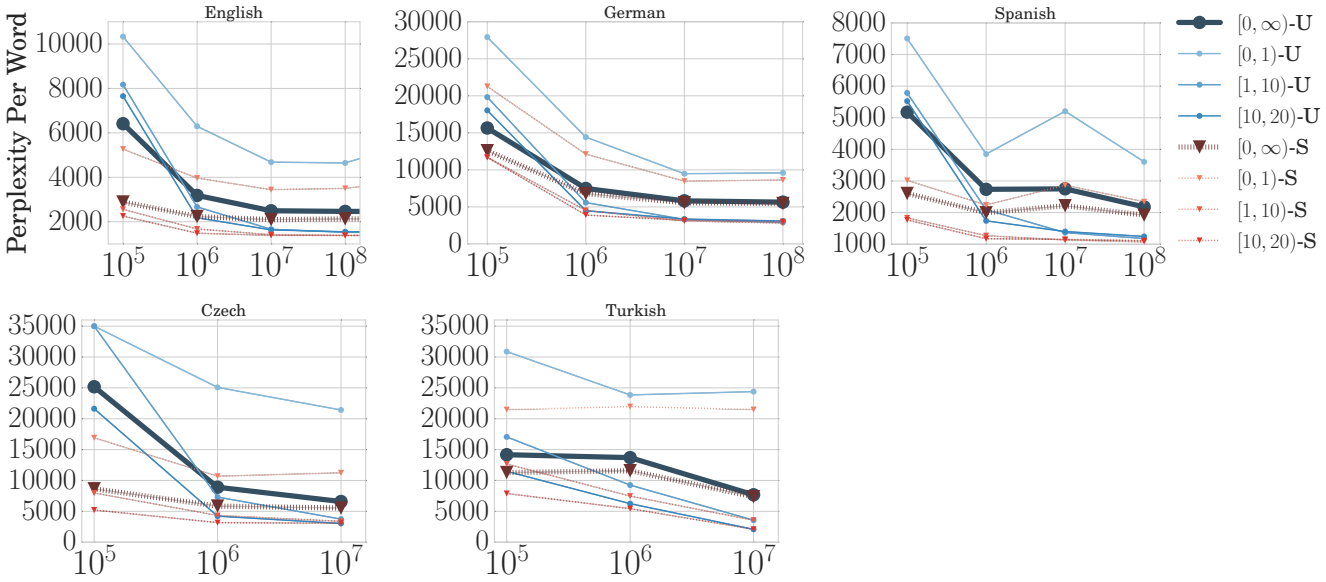

Figure 4: A full version of Figure 3, with all 5 languages.

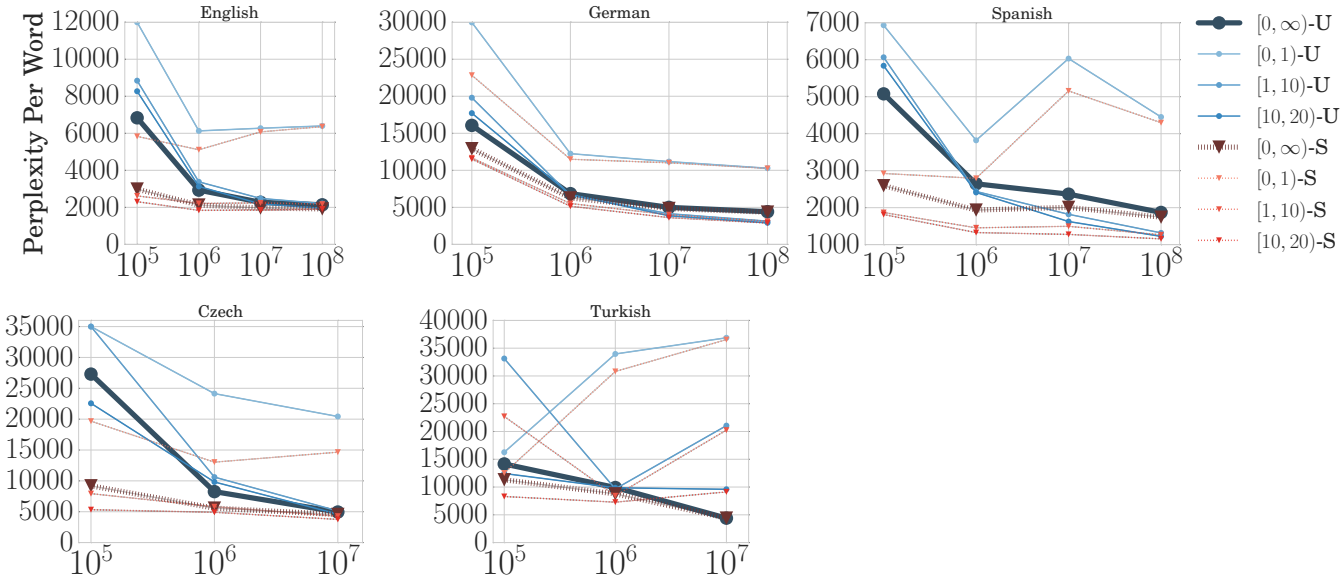

Figure 5: An alternate version of Figure 4. The aggregate curves are the same as before, but the frequency breakdown into word categories is now performed separately at each training size. These points are useful to look up or compare the performance of different word categories (novel, rare, frequent) at a given training size. However, the points along a given curve are incomparable: the $[0, 1)$ curve aggregates over many fewer types at the right than at the left. Sometimes all breakdown curves get worse at once even while their aggregate gets better, an example of Simpson's paradox.

C Coordinate Descent Algorithm for MAP Inference

We now derive the algorithm for maximizing the posterior probability $p(\boldsymbol{w}, \boldsymbol{m} \mid \boldsymbol{v})$. This is equivalent to minimizing (7), which is the negative log of the posterior probability plus a constant, repeated here:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{m}) = \sum_k \lambda ||m_k||_2^2 + \sum_i ||w_i - \sum_{k \in M_i} m_k||_{\Sigma_i}^2 + \sum_i ||v_i - w_i||_{\Sigma_i'/n_i}^2 \tag{9}$$

Recall that $||\boldsymbol{x}||_\Sigma^2 \stackrel{\text{def}}{=} \boldsymbol{x}^T \beth \boldsymbol{x}$ where $\beth \stackrel{\text{def}}{=} \Sigma^{-1}$, the inverse covariance matrix.

We take the partial gradient with respect to a particular vector m_k (renaming the dummy variable k to j):

$$\frac{\partial \mathcal{L}}{\partial m_k} = \frac{\partial}{\partial m_k} \left[\sum_j \lambda ||m_j||_2^2 + \sum_i ||w_i - \sum_{j \in M_i} m_j||_{\Sigma_i}^2 + \sum_i ||v_i - w_i||_{\Sigma_i'/n_i}^2 \right] \tag{10}$$

$$= \frac{\partial}{\partial m_k} \left[\sum_j \lambda ||m_j||_2^2 + \sum_i ||w_i - \sum_{j \in M_i} m_j||_{\Sigma_i}^2 \right] \tag{11}$$

$$= 2\lambda m_k. + \sum_{i \in W_k} -2\beth_i (w_i - \sum_{j \in M_i} m_j) \tag{12}$$

We now set this equal to $\mathbf{0}$:

$$\mathbf{0} = \lambda m_k - \sum_{i \in W_k} \beth_i (w_i - \sum_{j \in M_i} m_j) \tag{13}$$

Rearranging terms,

$$\lambda m_k = \sum_{i \in W_k} (\beth_i w_i - \sum_{j \in M_i} \beth_i m_j) \tag{14}$$

$$\lambda m_k + \sum_{i \in W_k} \beth_i m_k = \sum_{i \in W_k} (\beth_i w_i - \sum_{j \in M_i, j \neq k} \beth_i m_j) \tag{15}$$

$$\left(\lambda I + \sum_{i \in W_k} \beth_i \right) m_k = \sum_{i \in W_k} (\beth_i w_i - \sum_{j \in M_i, j \neq k} \beth_i m_j). \tag{16}$$

Finally, we arrive at the update rule for m_k:

$$m_k \leftarrow \left(\lambda I + \sum_{i \in W_k} \beth_i \right)^{-1} \sum_{i \in W_k} \beth_i (w_i - \sum_{j \in M_i, j \neq k} m_j). \tag{17}$$

Now take the partial gradient of $\mathcal{L}$ with respect to a particular vector w_i (renaming dummy variable i to j):

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial}{\partial w_i} \left[\sum_k \lambda ||m_k||_2^2 + \sum_j ||w_j - \sum_{k \in M_j} m_k||_{\Sigma_j}^2 + \sum_j ||v_j - w_j||_{\Sigma_i'/n_i}^2 \right] \tag{18}$$

$$= \frac{\partial}{\partial w_i} \left[\sum_j ||w_j - \sum_{k \in M_j} m_k||_{\Sigma_j}^2 + \sum_j ||v_j - w_j||_{\Sigma_i'/n_i}^2 \right] \tag{19}$$

$$= 2\beth_i (w_i - \sum_{k \in M_i} m_k) - 2n_i \beth_i' (v_i - w_i) \tag{20}$$

Setting this equal to $\mathbf{0}$, we get

$$\Im_i(w_i - \sum_{k \in M_i} m_k) = n_i \Im_i'(v_i - w_i) \tag{21}$$

$$\left(n_i \Im_i' + \Im_i\right) w_i = n_i \Im_i' v_i + \Im_i \sum_{k \in M_i} m_k \tag{22}$$

This yields the update rule

$$w_i \leftarrow (n_i \Im_i' + \Im_i)^{-1} \left(n_i \Im_i' v_i + \Im_i \sum_{k \in M_i} m_k \right) \tag{23}$$